(/)

Articles (/articles/) Cloud Storage (/cloud-storage) Business VoIP (/business-voip) Browse (/directory) Blog (/blog) utm_source=sourceforge&utm_medium=navbar&utm_campaign=homepage) ▾

Home (/) / Browse (/directory) / asfer (/p/asfer/) / Code

asfer (/p/asfer/)

Brought to you by: ka_shrinivaasan (/u/userid-769929/)

[r2012] (/p/asfer/code/2012/): (./../) / asfer-docs (./) / AstroInferDesign.txt Maximize (#) History (/p/asfer/code/2012/log/?path=/asfer-docs/AstroInferDesign.txt)

Download this file (?format=raw)

11417 lines (9736 with data), 1.4 MB

```
#-----------------------------------------------------------------------------------------------
#ASFER - Software for Mining Large Datasets
#This program is free software: you can redistribute it and/or modify
#it under the terms of the GNU General Public License as published by
#the Free Software Foundation, either version 3 of the License, or
#(at your option) any later version.
#This program is distributed in the hope that it will be useful,
#but WITHOUT ANY WARRANTY; without even the implied warranty of
#MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#GNU General Public License for more details.
#You should have received a copy of the GNU General Public License
#along with this program.  If not, see <http://www.gnu.org/licenses/>.
#-----------------------------------------------------------------------------------------------
#Copyleft (Copyright+):
#Srinivasan Kannan (alias) Ka.Shrinivaasan (alias) Shrinivas Kannan
#Ph: 9791499106, 9003082186
#Krishna iResearch Open Source Products Profiles:
#http://sourceforge.net/users/ka_shrinivaasan,
#https://github.com/shrinivaasanka,
#https://www.openhub.net/accounts/ka_shrinivaasan
#Personal website(research): https://sites.google.com/site/kuja27/
#emails: ka.shrinivaasan@gmail.com, shrinivas.kannan@gmail.com,
#kashrinivaasan@live.com
#-----------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------
NeuronRain - Design Objectives
-----------------------------------------------------------------------------------------------
NeuronRain OS is machine learning, cloud primitives enriched linux-kernel fork-off with applications in Internet-of-T
the machine learning side of it to learn analytics variables later read as config by linux kernel and any device spec
NeuronRain has been partitioned into components so that effective decoupling is achieved e.g Linux kernel can be real


Copyright attributions for other open source products dependencies:
----------------------------------------------------------------
1.Maitreya's Dreams - http://www.saravali.de (some bugs were fixed locally in degree computation of textual display m
2.SVMLight -  http://svmlight.joachims.org/
3.BioPython and ClustalOmega Multiple Sequence Alignment BioInformatics Tools (www.biopython.org, www.ebi.ac.uk/Tools

---------------------------
Open Source Design and Academic Research Notes have been uploaded to http://sourceforge.net/projects/acadpdrafts/file

Presently a complete string sequence mining subsystem and classification algorithms with indexing have been implement
***********************************************************************************************
                      AstroInfer Classifiers - Documentation on the dataset files used
***********************************************************************************************

decisiontree-attrvalues.txt - Attributes based on which decision is done and the values they can take (comma separate
decisiontree-test.txt - Test dataset with set of values for attributes in each line
decisiontree-training.txt - Training dataset with set of values for attributes and class it belongs in each line
test-set.txt - List of article id(s) to be classified - NaiveBayes
topics.txt - List of classes the article id(s) in training set belong to - NaiveBayes
training-set.txt - List of articles id(s) already classified - training dataset for NaiveBayes
word-frequency.txt - Words and their frequencies of occurrence in all articles - NaiveBayes
words.txt - words, the article id(s) having those words and number of occurrences of those words within specific arti

Above XXX.txt files need to be populated after doing some preprocessing of the articles to be classified. Preprocessi
```

```
58
59  Python script for autogenerating above txt files has been added under python-src/autogen_classifier_dataset.This scri
60
61  ********************************************************************************************************************
62  DESIGN NOTES, THEORETICAL INTERLUDES(might have errors), TODO AND NICE TO HAVE FEATURES (list is quite dynamic and mi
63  ********************************************************************************************************************
64
65  (FEATURE - DONE-MDL,Entropy,Edit Distance,Compressed Sensing) 1. Test with Massive Data Sets of encoded strings for p
66
67  (FEATURE - THEORY-POC Implementation-DONE) 2. An experimental text compression algorithm that deletes vowels and stor
68                          x1--------x2--------x3------x4
69                          |         |         |       |
70                          t         h         -       t
71  and Bayesian for the above is:
72          Pr(x3/[t,h,-,t]) is directly proportional to Pr(x3/[t,h]) * Pr([t]/x3)
73  and to be precise it is:
74          Pr(x3/[t,h,-,t] = Pr(x3/[t,h]) * Pr([t]/x3) / Pr(t)/Pr(t,h) with denominator being a pre-computable constant
75
76  Viterbi path would give the most likely path or most likely word for above.
77
78  (2.1) Pr(x3/[t,h]) is computed from dictionary - number of words having the sequence / total number of words
79  which is the argmax( Pr(a/t,h), Pr(e/t,h), Pr(i/t,h), Pr(o/t,h), Pr(u/t,h)) and priors are computed for the substring
80
81  (2.2) Pr(t/x3) is computed from dictionary similar to the previous for argmax of probabilities for substrings at, et,
82
83  (2.3) If total number words in dictionary is E (could be 200000 to 300000) and yi is the number of words of length i,
84
85  (2.4) Number of substrings of an n bit word is (n-1)n/2.
86
87  (2.5) Thus number of substrings for all words in dictionary is y2 + 3*y3 + 6*y4 + 10*y5 + ... + (n-1)n*yn/2. Thus num
88
89  (2.6) Above is theoretical basis only implementation of which needs precomputing the above hashtable.
90
91  (2.7) Above experimental compression scheme that ignores vowels gives a string complexity measure (a minimum descript
92
93  (2.8) Algebraically, above can be imagined as a Group Action where group G is set of consonants (with the assumption
94
95  (2.9) Linguistic words can be construed as a polynomial graph plotted on an x-y axis: x-axis represents position of a
96
97  (2.10) From point 345 and 347, high level language texts like English,French etc., can be thought of as 255-coloring
98
99  (2.11) Monochromatic arithmetic progressions in letter positions of natural language texts implies that finite state
100                         a---b---a---a--->b
101                                 |<-------|
102
103 (2.12) In terms of vowelless string complexity measure previously, if a 255-colored string is simplified to 2-colored
104
105 3. (FEATURE - THEORY - Pairwise LCS implementation DONE) KMeans, KNN and other clustering and classification algorith
106
107 4. (FEATURE - DONE) Implementation of String Distance measure using different distance measures (python-src/StringMat
108
109 5. (FEATURE - DONE) Construct a Hidden Markov Model with State transitions and Observations(events). If the number of
110
111 6. (FEATURE - DONE - continuation of point 3) Correlate with Rules from Classics (BPHS, BJ etc.,) with the mined data
112
113 7. (FEATURE - DONE) Integrate Classifiers in above String Pattern mining (classify strings and then mine).
114
115 8. (FEATURE - DONE) At present fundamental algorithms like - Perceptrons with Gradient Descent, Linear and Logistic R
116
117 9. (FEATURE - DONE) InterviewAlgorithm code in https://sourceforge.net/p/asfer/code/146/tree/python-src/InterviewAlgo
118
119 [Disclaimer and Caveat: Due to radical conclusions that are derivable from the P(good) binomial coefficients summatic
120
121 There are zero error voting systems:
122 E.g Paxos protocol without byzantine failures - http://research.microsoft.com/en-us/um/people/lamport/pubs/paxos-simp
123 ]
124 ---------------------------------------------------------------------------
125 Psuedorandom Choice and Majority Voting (Majority circuit with SAT inputs)
126 ---------------------------------------------------------------------------
127 10. (THEORY) https://sites.google.com/site/kuja27/ImplicationRandomGraphConvexHullsAndPerfectVoterProblem_2014-01-11.
128
129 Additional References:
130 ----------------------
131 10.1. Social Choice Theory, Arrow's Theorem and Boolean functions - http://www.ma.huji.ac.il/~kalai/CHAOS.pdf
132 10.2. http://www.project-syndicate.org/commentary/kenneth-arrow-impossibility-theorem-social-welfare-by-amartya-sen-2
133 10.3. Real life illustration of Arrow's Theorem - http://www.nytimes.com/2016/05/09/upshot/unusual-flavor-of-gop-prim
134 10.4. How hard is it to control elections [BartholdiToveyTrick] - http://www.sciencedirect.com/science/article/pii/08
135 10.5. How hard is it to control elections with tiebreaker [MatteiNarodytskaWalsh] - https://arxiv.org/pdf/1304.6174.p
136
137
138 11.(THEORY) What is perplexing is the fact that this seems to contravene guarantee of unique restricted partitions de
```

12.(THEORY) Moreover Arrow's theorem for 3 candidate condorcet election implies a non-zero probability of error in vc

13.(THEORY) But it is not known that if Arrow's theorem can be generalized for infinite number of candidates as above

14.(THEORY) Each Voter has a k-SAT circuit which is input to Majority circuit. k-SAT can be reduced to 3-SAT (not 2-S

For example, a term in the binomial summation becomes:
mC(n+1) * (1/q^s)^(n+1) * (1-1/q^s)^(m-n-1) where m is total number of voter SATs and (1/q) is probability of single

The Voter SAT circuit has an interesting application of Hastad's Switching Lemma - random probabilistic setting of va

As done in the Switching Lemma proof, parity circuit can be an AND of ORs (or OR of ANDs) which is randomly restricte

Fourier expansion of the unbounded fan-in Majority+SAT circuit is obtained from indicator polynomial (http://www.cont

Demarcation of linear and exponential sized ciruits: P(good) RHS has Circuit SAT for each voter with unbounded fanin.
1) If the variables for each voter SAT are same, then the circuit size is exponential in number of variables - DC-uni
2) else if the variables are different for each SAT, then the circuit is linear in number of variables - polynomial s

1) is more likely than 2) if lot of variables among voters are common.

There are three possibilities
Circuit lies between 1) and 2) - some variables are common across voters - kind of super-polynomial and sub-exponenti
Circuit is 1) - all variables are common across voters - the worst case exponential (or even ackermann function?)
Circuit is 2) - there are no common variables

As an alternative formulation, LHS of the P(Good) series can be a dictator boolean function (property testing algorit

Influence of a variable i in a boolean function is Probability[f(x) != f(x with i-th bit flipped)]. For Majority func

As done for sensitivity previously, a term in the P(Good) binomial summation becomes:
mC(n+1) * product_of_n+1_(NoiseSensitivityOfVoterSAT(i)) * product_of_m-n-1_(1-NoiseSensitivityOfVoterSAT(i)) where m

Summation of binomial coefficients in RHS is the collective stability or noise in the infinite majority circuit. It c

---
P(Good) summation is Complementary Cumulative Binomial Distribution Function (or) Survival Function
---
Goodness Probability of an elected choice is derived in https://sites.google.com/site/kuja27/CircuitForComputingError

A plausible proof for the assumption that for an above average good choice atleast half of the popluation must have m
If there are n inputs to the Majority circuits, the voter inputs with good decision can be termed as "Noiseless" and
        lim NS(delta, Majority(n)) = 1/pi * arccos(1-2*delta) (due to Central Limit Theorem and Sheppard's Formula)
        n->infinity
        n odd

where delta is probability of noise and hence bad decision. If delta is assumed to be 0.5 for each voter (each voter
        1/pi * arccos(1-1) = 1/pi * pi/2 = 1/2 which is the Noise (Goodness or Badness) of the majority circuit as a
From this expected number of bad decision voters can be deduced as E(x) = x*p(x) = 0.5 * n = n/2 (halfway mark)

Above is infact a simpler proof for P(Good) without any binomial summation series for delta=0.5. In other words the n

For delta=0.5, NoiseSensitivity of infinite majority=0.5 and both are equal. For other values of delta this is not th

delta=0.3333...:

NS = 1/pi * arccos(1 - 2*0.333...) = 0.3918

delta=0.25:

NS = 1/pi * arccos(1-2*0.25) = 0.333...

Chernoff upper tail bound

Upper bound for P(Good) binomial distribution can be derived from Chernoff bound (assuming Central Limit Theorem appl
        P[X > (1+delta)*mean] < {e^delta/(1+delta)^(1+delta)}^mean
        Total population is n, mean=n*p and (1+delta)*mean = n/2
        Thus delta = (1 - 2*p)/2*p
        P[X > n/2] < {e^[(1-2p)/p]/(1/2p)^(1/2p)} = {e^(1-2p)n * (2p)^(1/2p)}

When p=0.5 , P[X > n/2] < 1 which is an upperbound for 0.5.
But intriguingly for p=1 the upper tail bound is:
                (e^(-1)*sqrt(2))^n
which is increasingly less than 1 for high values of n whereas P(good) converges to 1 in exact case (why are these di
    mC(n+1)(p)^(n+1)(1-p)^(m-n-1) + mC(n+2)(p)^(n+2)(1-p)^(m-n-2) + ... + mCm (p)^m (1-p)^(m-m)
all the terms vanish by zero multiplication except mCm (p)^m (1-p)^m which becomes:
        mCm*1*0^0
and by convention, 0^0 = 1, thereby converging to 1 while the Chernoff upper tail bound decreases with n.

```
220
221  -------------------------
222  Hoeffding upper tail bound
223  -------------------------
224  For Bernoulli trials involving binomial distribution summation, Hoeffding Inequality can be applied for upper tail bo
225          P(X > (p+epsilon)n) <= exp(-2*epsilon^2 * n)
226
227  If p+epsilon = 0.5 for halfway mark, epsilon = 1/2 -p.
228  P(X > n/2) <= exp(-2(0.5-p)^2*n)
229  For p=1, P(X > n/2) <= exp(-n/2) which for infinite n converges to 1.
230
231  Thus tailbounds above are only less tight estimates and exact summation is better for p=0.5 and 1.
232
233  ---------------------------------------------------------------------------
234  There are two independent aspects to the P(Good) series - Goodness and Hardness:
235  ---------------------------------------------------------------------------
236  (1) Goodness of Voting:
237  The convergence of the binomial coefficient summation series in RHS is the "goodness" side of the voting expressed in
238  (2) Hardness of Voting:
239  The circuit for RHS(e.g Boolean Function Composition) and its Fourier polynomial expansion.
240
241  Mix of Hardness and Goodness result in variety of BP* classes or P* classes (if error is unbounded). Goodness is rela
242
243  ------------------------------------------------------------------------------------------------------------------
244  (*IMPORTANT*) Noise Stability, Noise Sensitivity and Complexity Class BPP (error in voting)
245  ------------------------------------------------------------------------------------------------------------------
246  In all points in this document, an assumption is made that Noise Stability and BPP are equivalent notions. Is this va
247  ---------------------------------------------------------------------------------------
248          x                               |    f(x) = f(x/e)      |       f(x) != f(x/e) Noise    |
249  ---------------------------------------------------------------------------------------
250  x in L, x/e in L                        |       No error        |       Error                   |
251  ---------------------------------------------------------------------------------------
252  x in L, x/e not in L                    |       Error           |   No error if f(x)=1,f(x/e)=0 |
253                                          |                       |   else Error                  |
254  ---------------------------------------------------------------------------------------
255  x not in L, x/e in L                    |       Error           |   No error if f(x)=0,f(x/e)=1 |
256                                          |                       |   else Error                  |
257  ---------------------------------------------------------------------------------------
258  x not in L, x/e not in L                |       No error        |      Error                    |
259  ---------------------------------------------------------------------------------------
260
261  Third column of the table relates Noise Sensitivity and BPP - it subdivides the Noise Sensitivity into 4 probable sce
262          Probability of Good Decision = Input noise stability (second column) + Output noise stability (third column)
263  which can also be stated as:
264          Probability of Good Decision = 1-(Probability of False positives + Probability of False negatives)
265
266  Above is a better ,rather ideal, error probability measure that can be substituted in P(Good) RHS binomial distributi
267          Probability of Good Decision = Pr(f(x)=f(x/e) / x in L, x/e in L)*Pr(x in L, x/e in L) +
268                                         Pr(f(x)!=f(x/e) / x in L, x/e in L)*Pr(x in L, x/e in L) +
269                                          ...
270                                         Pr(f(x)!=f(x/e) / x not in L, x/e not in L)*Pr(x not in L, x/e not in L)
271  Computation of the above conditional probability summation further confounds the estimation of voter decision error i
272
273  In other words, a precise estimate of voter or voting error from table above which related BP* complexity class and E
274          Probability of Bad Decision =  NoiseSensitivity - {Pr(f(x)=1, f(x/e)=0)/x in L, x/e not in L}
275                                                         - {Pr(f(x)=0, f(x/e)=1)/x not in L, x/e in L}
276                                                         + {Pr(f(x)=f(x/e))/x in L, x/e not in L}
277                                                         + {Pr(f(x)=f(x/e))/x not in L, x/e in L}
278  which may be more or less than NoiseSensitivity depending on cancellation of other conditional probability terms. Thi
279  has to be substituted for each voter decision making error. Because of this delta, NoiseSensitivity (and NoiseStabili
280
281  When Probability of Bad Decision is 0 across the board for all voters, the Pr(Good) series converges to 100% and prev
282         NoiseSensitivity = {Pr(f(x)=1, f(x/e)=0)/x in L, x/e not in L}
283                            {Pr(f(x)=0, f(x/e)=1)/x not in L, x/e in L}
284                            - {Pr(f(x)=f(x/e))/x in L, x/e not in L}
285                            - {Pr(f(x)=f(x/e))/x not in L, x/e in L}
286
287  Above matrix of 8 scenarios and identity above give somewhat a counterintuitive and startling possibility that NoiseS
288
289  ------------------------------------------------------------------------------------------------
290  Hastad Switching Lemma, P(Good) Majority Voting circuit and Election Approximation or Forecasting
291  ------------------------------------------------------------------------------------------------
292  Hastad Switching Lemma for circuit of width w and size s with probability of random restriction p for each variable s
293     Pr[DecisionTree(f/restricted) > k ] <= (constant*p*w)^k
294  where p is a function of width w and p is proportional to 1/w. In forecasting, a sample of voters and their boolean f
295  (1) Pseudorandomly choose subset of Voter Boolean Functions in the first layer of the circuit - This is also a random
296  (2) Randomly Restrict variables in each boolean circuit to get decision trees of required depth if voters reveal thei
297  (3) Simulated Voting (and a property testing) on the restricted boolean functions subset accuracy of which is a condi
298
299  (1) and (2) are random restrictions happening in 2 different layers of the circuit - amongst voters and amongst varia
300
```

```
Alternatively if subset of voters reveal their votes (1 or 0) in opinion polls then the above becomes a Learning Thec

An interesting parallel to election approximation is to sample fourier spectrum of boolean functions of a complexity

Points (53.1) and (53.2) define a Judge Boolean Function with TQBF example that is PSPACE-complete. Each voter in RHS

-----------------------------------------------------------------------------------------------
(*IMPORTANT*) Boolean Function Composition (BFC) and Majority+SAT voting circuit
-----------------------------------------------------------------------------------------------
Boolean Function Composition is described in detail at [AvishayTal] Section 2.3 - http://eccc.hpi-web.de/report/2012/


[
Quoting Corollary 6.2 in http://eccc.hpi-web.de/report/2012/163/ for inclusions of complexity measures:
                "... C(f) ��� D(F) ��� bs(f) ��  deg(f) ��� deg(f)^3 ..."
]

-----------------------------------------------------------------------------------------------
Noise Sensitivity and Probability of Goodness in P(Good) LHS and RHS - Elaboration on previous
-----------------------------------------------------------------------------------------------
LHS of P(Good):
--------------
(1) Depends either on a natural (psuedo)random process or a Dictator Boolean Function that unilaterally decides outcc

(2) When error is zero, probability of good decision is 1 (or) Pr(f(x) != f(y)) is zero where x and y are correlated
a bit flip. In other words Noise in input does not alter outcome. If NS is Noise Sensitivity, probability of good out

RHS of P(Good):
---------------
(1) Depends on the Noise Sensitivity of Boolean Function Composition - Maj(n)*SAT - Majn(SAT1, SAT2,....,SATn).

(2) When error is zero, probability of good decision is 1 because nCn*(1)^n*(1-1)^(0) becomes 1 in summation and all

(3) NoiseSensitivity is a fitting measure for voter error or voting error as it estimates the probability of how a ra

(4) Perfect Voter has a SAT or Boolean Function that is completely resilient to correlation i.e Pr(f(x) != f(y)) = 0.

(5) Brute Force exponential algorithm to find perfect voter is by property testing:
   - For 2^n bit strings find correlation pairs with random flips - 2^n * (2^n - 1) such pairs are possible.
   - Test Pr(f(x) != f(y))
   - This is not a learning algorithm.

(6) For perfect voter, probability p of good decision is 1-NS or 1/2+1/2*Stability and has to be 1 which makes NS=0 a

(7) Open Question: Is there a learning algorithm that outputs a Perfect Voter Decision Boolean Function f such that N

Is there some other function stabler than stablest - answer has to be no - But there is an open "Majority is the Leas
For Linear Threshold Functions (majority, weighted majority et al) f, Stability(f) >= Stability(Majn).
Majority is Stablest theorem is for small-influence LTFs (maximum influence of f < epsilon). If existence of perfect

An obvious corollary is that for stablest voting boolean function, majority is apt choice (or) in the Majority voting

(8) Theorem 2.45 in http://analysisofbooleanfunctions.org/ is important which proves Stability and NoiseSensitivity t
    lt (n->infinity, n odd) Stability[Majn] = 2/pi*arcsin(rho)
    (or)
    lt (n->infinity, n odd) NS[delta,Majn] = 2/pi*delta + O(delta^1.5) where delta is the probability of correlation

(9) There are two random variables f(x)=f(y) and f(x) != f(y) with respective probabilities.

(10) This implies for infinite electorate, error probability is (an error that causes x to become y with probability
    lt (n->infinity, n odd) NS[delta,Majn] = 2/pi*delta + O(delta^1.5) where delta is the probability of correlation

(11) probability of good outcome = p = 1-NS:
    p = 1 - lt (n->infinity, n odd) NS[delta,Majn] = 1 - 2/pi*delta - O(delta^1.5).

(12) If delta = 1:
    lt (n->infinity, n odd) NS[delta,Majn] = 2/pi + O(1).
    (and)
    probability of good outcome = 1-NS = 1/2+1/2*Stability = 1-2/pi-O(1).

(13) In P(Good) RHS:
--------------------
If the error probability of voter is 0 and probability of good decision is 1 then binomial summation becomes nCn*(p)^

(14) In P(Good) LHS:
--------------------
If error probability of dictator boolean function is 0 and p=1 then:
 p = 1 - NS(DictatorFunction)
   = 1 - delta^n
Similar to RHS p=1 iff delta=0
Thus if there is no Noise on either side LHS=RHS (or) LHS is a P algorithm to RHS EXP or PH=DC. This depends on exist
```

(15) [Benjamini-Kalai-Schramm] theorem states the condition for a boolean function to be noise sensitive:
A boolean function is noise sensitive if and only if L2 norm (sum of squares) of influences of variables in f tends t
Therefore, for stability any voter boolean function in P(Good) RHS should not have the L2 norm of influences to tend

(16) "How much increasing sets are positively correlated" by [Talagrand] - http://boolean-analysis.blogspot.in/2007/6
Weight(f) = sum_1_to_n(fouriercoeff(i)) <= O(mean^2*1/log(mean))
where Weight of boolean function f in n variables is the sum of squares of n fourier coefficients and mean is Pr(f(x)

(17) By Majority is Stablest Theorem if P(Good) RHS is a boolean function composition of Maj*Maj=Majn(Maj1,Maj2,...,M
(1-2/pi * arccos(rho))*(1-2/pi * arccos(rho)) assuming that stability of the composition is the product like other
If the LHS of P(Good) is the Dictator Boolean Function its stability is (rho)^n (because all bits have to be flipped)
[cos(pi/2*(1-(rho)^n))]^2 = rho

For rho=0:

[cos (pi/2*(1-0^n))]^2 = 0 hence LHS=RHS

For rho=1:

[cos (pi/2*(1-1^n))]^2 = 1 hence LHS=RHS

But for rho=0.5:
---
arccos 1/sqrt(2) = pi/2*(1-0.5^n)
n = log(1-2/pi*0.7071)/log(0.5)
n = 0.863499276 which is quite meaningless probably hinting that for uniform distribution the parity size n is < 1.

(18) In RHS the P(Good) binomial coefficient summation in terms of Noise Sensitivities of each Voter SAT:
nC0(1-NS)^0(NS)^n + nC1(1-NS)^1(NS)^(n-1) + nC2(1-NS)^2(NS)^(n-2) + ...
Above summation should ideally converge to holistic Noise Sensitivity of the Boolean Function Composition of Maj*Maj
1/2 + 1/2*(1-2/pi*arccos(rho))^2 = nC0(1-NS)^0(NS)^n + nC1(1-NS)^1(NS)^(n-1) + nC2(1-NS)^2(NS)^(n-2) + ...

If each voter has boolean function with varied Noise Sensitivities, above becomes a Poisson Trial instead of Bernoull

(19) There are classes of boolean functions based on percolation crossings (http://arxiv.org/pdf/1102.5761.pdf - Noi

For epsilon=o(1/n^2*alpha4), Pr[fn(sequence)=fn(sequence with epsilon perturbation)] tends to zero at infinity where

Open question: If Percolation Boolean Functions are 100% noise stable (as in infinite grid previously), what is the l

(*IMPORTANT* - this requires lot of reviewing - if correct places a theoretical limit on when stability occurs) From

(20) Noise stability in terms of Fourier coefficient weights is defined as:
summation(rho^k * W^k) for all degree-k weights where W = summation(fourier_coeff(S)^2) for all |S|, S in [

(21) Variant 1 of Percolation as a Judge boolean function is defined in 53.4 below which is in non-uniform NC1. If th

(22) Also from Theorem 5.17 - [RyanODonnell] - http://analysisofbooleanfunctions.org/ - for any rho in [0,1], Stabil
2/pi*arcsin ?? ??? Stab??[Majn] ??? 2/pi*arcsin ?? + O(1/(sqrt(1-rho^2)*sqrt(n))).

(23) Peres' Theorem rephrases above bound for NoiseStability in terms of NoiseSensitivity for class of uniform noise
NoiseSensitivity??[f] ??? O(sqrt(??))

(24) From http://arxiv.org/pdf/1504.03398.pdf: [Boppana-Linial-Mansour-Nisan] theorem states that influence of a bool
Inf(f) = O((logS)^d-1)

(25) [Benjamini-Kalai-Schramm] Conjecture (different from Majority is least stable BKS conjecture which is still open
Every monotone boolean function f can be epsilon (e) approximated by a circuit of depth d and of size exp((K(e)*Inf(f

(26) Class of uniform noise stable LTFs in (23) are quite apt for Voter Boolean Functions in the absence of 100% nois

---
Additional References:

14.1 k-SAT and 3-SAT - http://cstheory.stackexchange.com/questions/7213/direct-sat-to-3-sat-reduction
14.2 k-SAT and 3-SAT - http://www-verimag.imag.fr/~duclos/teaching/inf242/SAT-3SAT-and-other-red.pdf
14.3 Switching Lemma
14.4 Donald Knuth - Satisfiability textbook chapter - http://www-cs-faculty.stanford.edu/~uno/fasc6a.ps.gz - Quoted e
" ... Section 7.2.2.2 Satisfiability
Satisfiability is a natural progenitor of every NP-complete problem???
Footnote: At the present time very few people believe that P = NP. In other words, almost everybody who has studied
14.5 Majority and Parity not in AC0 - www.cs.tau.ac.il/~safra/ACT/CCandSpaceB.ppt, http://www.math.rutgers.edu/~sk123
14.6 Sampling Fourier Polynomials - http://cstheory.stackexchange.com/questions/17431/the-complexity-of-sampling-appr

14.7 Kummer's theorem on binomial coefficient summation - quoted from Wikipedia:
"In mathematics, Kummer's theorem on binomial coeffients gives the highest power of a prime number p dividing a binom

14.8 Summation of first k binomial coefficients for fixed n:
- http://mathoverflow.net/questions/17202/sum-of-the-first-k-binomial-coefficients-for-fixed-n
- https://books.google.co.in/books?id=Tn0pBAAAQBAJ&pg=PA61&lpg=PA61&dq=summation+of+first+k+binomial+coefficients&so

```
which imply bounds for the summation:
 - sum_k_2n(2nCi) = 2^2n - sum_0_k-1(2nCi)
 - 2^2n - sum_0_k-1(2nCi) >  2^2n - (2nCk) * 2^(2n) / 2*(2nCn)

14.9 Probabilistic boolean formulae - truth table with probabilities - simulates the Voter Circuit SAT with errors
http://www.cs.rice.edu/~kvp1/probabilisticboolean.pdf

14.10 BPAC circuits ( = Probabilistic CNF?)
http://www.cs.jhu.edu/~lixints/class/nw.pdf

14.11 Mark Fey's proof of infinite version of May's theorem for 2 candidate majority voting - Cantor set arguments fo
https://www.rochester.edu/college/faculty/markfey/papers/MayRevised2.pdf

14.12 Upperbounds for Binomial Coefficients Summation - http://mathoverflow.net/questions/17202/sum-of-the-first-k-bi
           f(n,k)��  2^(n���1)*exp((n���2k���2)^2/4(1+k���n))
for f(n,k) = sum_i_0_to_k(nCi). Proof by Lovasz at: http://yaroslavvb.com/upload/lovasz-proof2.pdf

-----------------------------------------------------------------------------------
14.13 (*IMPORTANT*) Connections between Hypergeometric series and Riemann Zeta Function :
-----------------------------------------------------------------------------------
- http://matwbn.icm.edu.pl/ksiazki/aa/aa82/aa8221.pdf - This relation implies that:
      - P(good) binomial coefficient infinite series summation (and hence the Psuedorandom Vs Majority Voting circuit
      - and Complement Function circuit special case for RZF (and hence the Euler-Fourier polynomial for Riemann Zeta
are deeply intertwined. Kummer theorem gives the p-adic number (power of a prime that divides an integer) for dividin

Above implies that binomial coefficients can be written in terms of prime powers and vice-versa: P(good) series can b

-------------------------------------------------------------------------
14.15 ACC circuits and P(Good) RHS circuit
-------------------------------------------------------------------------
ACC circuits are AC circuits augmented with mod(m) gates. i.e output 1 iff sum(xi) is divisible by m. It is known tha

-----------------------------------------------------
14.16. Infinite Majority as Erdos Discrepancy Problem
-----------------------------------------------------
Related to (132) - May's Theorem proves conditions for infinite majority with sign inversions {+1,-1} and abstention

14.17 Noise Sensitivity of Boolean Functions and Percolation - http://arxiv.org/pdf/1102.5761.pdf, www.ma.huji.ac.il/

14.18 Binomial distributions, Bernoulli trials (which is the basis for modelling voter decision making - good or bad

14.19 Fourier Coefficients of Majority Function -  http://www.contrib.andrew.cmu.edu/~ryanod/?p=877,  Majority and Ce

14.20 Hastad's Switching Lemma - http://windowsontheory.org/2012/07/10/the-switching-lemma/, http://www.contrib.andre

14.21 Degree of Fourier Polynomial and Decision tree depth - http://www.contrib.andrew.cmu.edu/~ryanod/?p=547#propdt-

14.22 Judgement Aggregation (a generalization of majority voting) - Arriving at a consensus on divided judgements - h

14.23 Noise and Influence - [Gil Kalai, ICS2011] - https://gilkalai.files.wordpress.com/2011/01/ics11.ppt

14.24 Doctrinal Paradox (or) Discursive dilemma - Paradoxes in Majority voting where both Yes and No are possible in

-----------------------------------------------------------

15. (FEATURE - DONE-Minimum Implementation using NetworkX algorithms) Add Graph Search (for example subgraph isomorph

16. (FEATURE - DONE) Use of already implemented Bayesian and Decision Tree Classifiers on astronomical data set for p

17. (FEATURE - DONE) Added Support for datasets other than USGS EQ dataset(parseUSGSdata.py). NOAA HURDAT2 dataset ha

18. (FEATURE - Minimum Functionality DONE-TwitterFollowersGraphRendering, Centrality) (Astro)PsychoSocialAnalysis  -
------------
References:
-----------
18.1 Networks,Crowds,Markets - http://www.cs.cornell.edu/home/kleinber/networks-book/networks-book.pdf
18.2 Introduction to Social Network Methods - http://faculty.ucr.edu/~hanneman/nettext/
18.3 Bonacich Power Centrality - http://www.leonidzhukov.net/hse/2014/socialnetworks/papers/Bonacich-Centrality.pdf
18.4 Modelling Human Emotions - http://www.jmlr.org/proceedings/papers/v31/kim13a.pdf
18.5 Depression Detection - http://www.ubiwot.cn/download/A%20Depression%20Detection%20Model%20Based%20on%20Sentiment
18.6 Market Sentiments - http://www.forexfraternity.com/chapter-4-fundamentals/sentiment-analysis/the-psychology-of-s
18.7 Sentiment Analysis - http://www.lct-master.org/files/MullenSentimentCourseSlides.pdf
18.8 Dream Analysis - http://cogprints.org/5030/1/NRC-48725.pdf
18.9 Opinion Mining - http://www.cs.uic.edu/~liub/FBS/SentimentAnalysis-and-OpinionMining.pdf
18.10 Linked - [Albert Laszlo Barabazi] - [Ginestra Bianconi - http://www.maths.qmul.ac.uk/~gbianconi/Condensation.ht
18.11 Small World Experiment and Six Degrees of Separation - [Stanley Milgram] - https://en.wikipedia.org/wiki/Small-
18.12 LogLog estimation of Degrees in Facebook graph - 3.57 Degrees of Separation - https://research.facebook.com/blo

19. (FEATURE - DONE-PAC Learnt Boolean Conjunction) Implement prototypical PAC learning of Boolean Functions from dat
        19.1 There are two datasets - dataset1 of size 2^n of consecutive integers and dataset2 of first n-bit prime
        19.2 Each element of dataset1 is mapped to i-th bit of the corresponding prime number element in the dataset2
```

```
        19.3 Above step gives i probabilistic, approximate, correct learnt boolean conjunctions for each bit of all t
        19.4 Reference: PAC Learning - http://www.cis.temple.edu/~giorgio/cis587/readings/pac.html
        19.5 PAC Learnt Boolean Conjunction is an approximated Decision Tree.
        19.6 PAC learning is based on Occam's Razor (no unnecessary variables)
PAC Learning implementation for learning patterns in Binary encoded first 10000 Prime Numbers have been committed to

20. (FEATURE - DONE) Integrate AstroInfer to VIRGO-KingCobra-USBmd Platform which makes it an Approximately Intellige
       20.1 Usual cloud platforms support only application-application layer RPCs over an existing operating system.
       20.2 NeuronRain differs in that aspect by adding new cloud features (with advanced features in development) li
       20.3 Cloud support within kernelspace has lot of advantages - low network latency because of kernelspace-kerne
       20.4 Kernelspace-Kernelspace communication is a privileged mode privy to kernel alone and userspace applicatic
       20.5 At present the data sent over kernel sockets is not encrypted which requires OpenSSL in kernel. Kernel dc
       20.6 There are ongoing efforts to provide Transport Layer Security (TLS) in kernel sockets which are not yet m
              20.6.1 KTLS - [DaveWatson - Facebook] - https://lwn.net/Articles/666509/ and https://github.com/ktls/a
              20.6.2 IPSec for kernel sockets - [SowminiVaradhan - Oracle] - http://www.netdevconf.org/1.1/proceedir
       20.7 Usual userspace-userspace network communication has following routing - user1----kernel1----kernel2----us

21. (FEATURE - DONE-Minimum Implementation) Unsupervised Named Entity Recognition and Part-of-Speech tagging using Co

###############################################################################################
A.      Design Notes on Mining Numerical Datasets
###############################################################################################

22. (FEATURE - DONE-Minimum Implementation) Period Three theorem (www.its.caltech.edu/~matilde/LiYorke.pdf) which is

23. (FEATURE - DONE) Entropy of numerical data - Sum of -Pr(x)LogPr(x) for all possible outcome probabilities for a r


#####################################################################################################################################

=====================================================================================================================================
(THEORY) DECIDABILITY OF EXISTENCE AND CONSTRUCTION OF COMPLEMENT OF A FUNCTION (important draft additions to http://
=====================================================================================================================================

(*)24. (DONE) COMPLEMENT OF A FUNCTION - Approximating or Interpolating with a Polynomial: This is quite expensive ar
        p(t) = 1/N [X0 + X1*e^(2*pi*i*t) + ... upto XN] and
        p(n/N) = xn
Thus DFT amd interpolation coincide in the above.

-------------------------------------------------------------------------------------------------------------------------------------
24a. Theorems on prime number generating polynomials - special case is to generate all primes or integral complement
-------------------------------------------------------------------------------------------------------------------------------------
Legendre - There is no rational algebraic function which always gives primes.
Goldbach - No polynomial with integer coefficients can give primes for all integer values
Jones-Sato-Wada-Wiens - Polynomial of degree 25 in 26 variables whose values are exactly primes exists

Polynomial interpolation and Fourier expansion of the boolean function in http://arxiv.org/pdf/1106.4102v1.pdf probab

-----------------------------------------------------------------------------------------
24b. Circuit construction for the complement function boolean DNF constructed in http://arxiv.org/pdf/1106.4102v1.pdf
-----------------------------------------------------------------------------------------
The DNF constructed for complement function has to be minimized (through some lowerbound techniques) before it is rep

Old draft of the complement function - https://sites.google.com/site/kuja27/ComplementOfAFunction_earlier_draft.pdf?a

Due to equivalence of Riemann Zeta Function and Euler's theorem - inverse(infiniteproduct(1-1/p(i)^z)) for all primes

Fourier polynomial of a boolean formula is of the form:
        f(x) = Sigma_S(fouriercoeff(S)parityfn(S))

and is multilinear with variables for each input. S is the restriction or powerset of the bit positions.

Thus if a prime number is b-bit, there are b fourier polynomials for each bit of the prime. Thus for x-th prime numbe
        P(x) = 1+2*fx1(x)+2^2*fx2(x).....+2^b*fxb(x)
where each fxi() is one of the b fourier expansion polynomials for prime bits.

-------------------------------------------------------------------------------------------------------------------------------------
24c. Riemann Zeta Function written as Euler-Fourier Polynomial :
-------------------------------------------------------------
Fourier polynomials for each prime can be substituted in Euler formula and equated to Riemann Zeta Function:
RZF = Inverse((1-1/P(x1)^s)(1-1/P(x2)^s))...ad infinitum-------------------------------------------------------------------

Non-trivial complex zero s is obtained by,
        P(xi)^s = 1
        P(xi)   = (1)^(1/s)
        1+2*fx1(xi)+2^2*fx2(xi).....+2^b*fxb(xi) = (1)^(e^(-i*theta)/r) after rewriting in phasor notation s=r*e^(i*t

Above links Fourier polynomials for each prime to roots of unity involving non-trivial zeroes of Riemann Zeta Functic

LHS is multilinear with at most b variables for b bits of the prime number. A striking aspect of the above is that Fc

```

If s is written as a+ic, then (1)^(1/s) = (1)^(1/a+ic) = (1)^((a-ic)/(a^2+c^2))

If RH is true a=0.5 and above becomes, (1)^((0.5-ic)/(0.25+c^2)).

1+2*fx1(xi)+2^2*fx2(xi).....+2^b*fxb(xi) = (1)^((0.5-ic)/(0.25+c^2)) ----------(3)

Thus imaginary part c has one-to-one correspondence to each prime. Non-trivial zeroes are usually found using functic

---

24d. Delving deeper into (1) above which equates Riemann Zeta Function with Complement Function Fourier Polynomials s

---

Finding pattern in distribution of primes is equivalent to finding pattern in above set of prime bit circuits or four

Similar to zeros of RZF, Fourier poynomial obtained from Euler's formula by substituting complement function prime bi

Important thing to note is that what it means to be a zero of the Euler-Fourier polynomial [(1),(2) and (3)] which ha

Sensitivity and Block sensitity of the above complement function circuit should also have a strong relation to prime

---

24e. Arithmetic Circuit for Euler Product and RZF - Alternative formulation to Euler-Fourier polynomials

---

Instead of a boolean circuit above can be an arithmetic circuit (with * and + gates with elements of some field as in

M(s) = | a -b|
| b a|

Thus p^s can be written as p^M(s) with matrix exponent. Rewriting this as e^(M(s) ln p) and expanding as series,

e^(M(s) ln p) = 1 + M ln p + M*M lnp^2/2! + ...
and 1 - 1/p^s = 1 - e^(-M(s) ln p) = 1-(1-M*ln p+ M*M ln p^2/2! - ...) = M*ln p - M*M ln p^2/2! + ... (alternatin

Thus each prime clause in the Euler product is written as an infinite summation of 2*2 matrix products. Using the tra

Instead of Euler formula, circuit for Riemann Zeta Function can be constructed with + gate at the root and circuits f
RZF(n) = n - M(s) * (ln 2 + ln 3 + ...+ ln n)/1! + M(s)^2 * (ln 2^2 + ln3^2 +...+ lnn^2)/2! + ... + M(s)^n *

The reason for drawing above arithmetic circuit is to represent the Riemann Zeta Function in complex plane as a circu

Using Taylor series expansion and Jordan Normal Form above can be written as a huge square matrix.(http://en.wikipedi

Eigen values of Chaotic systems Wave modelling Random matrices (matrix as random variable) have been shown to have a

---

24f. Can be ignored - quite Experimental - Different way to reduce the complex exponent in (1)

From (1), P(xi)^s = 1.
=> P(xi) ^ (k+il) = 1 where s=k+il
=> P(xi)^k * P(xi)^(il) - 1 = 0
P(xi)^k * [cos(l*ln(P(xi))) + i*sin(l*ln(P(xi)))] - 1 = 0 ----------(4)

Thus real and imaginary parts can be equated to zero independently as,
P(xi)^k * cos(l*ln(P(xi))) = 1 ----------(5)

P(xi)^k * sin(l*ln(P(xi))) = 0 ----------(6)

From (5) and (6), it can be deduced that:

tan(l*ln(P(xi))) = 0 ----------(7)

From (7), it can be inferred that Re(s)=k is not involved and PIT has to be done only on the b+1 variables (b bits in

tan(T) = T + T^3/3 + 2*T^5/15 + ... = 0 where T = l*ln(P(xi)) ---------(8)

l*ln(P(xi)) = 0 =>
l = 0 or ln(P(xi)) = 0
P(xi) = 1
both of which can not hold.

From (5),
P(xi)^k = sec(l*ln(P(xi))) ----------(9)

if T = l*lnP(xi), e^(T/l) = P(xi)

P(xi)^k = e^(kT/l) = 1+T^2/2 + 5T^4/24 + 61*T^6/720 + 277T^8/8064 + ... (expansion of sec) ---------- (1

1+kT/l + (kT/l)^2/2! + (kT/l)^3/3! + ... = 1 + T^2/2 + 5T^4/24 + ...(expansion of e^(KT/l)) --------(1

(5) can also be written as,

ln P(xi)^k = ln(sec(l*ln(P(xi)))

k = ln[sec(l*ln(P(xi)))] / lnP(xi) ----------(1

By RH k has to be 0.5 irrespective of RHS.

Approximation of l:

By assuming k=0.5 and Using series expansion of cos(x) - choosing only first two terms, l is approximately,

P(xi) = 1/[cos(l*lnP(xi))]^2 ----------(1
l = sqrt(2)*sqrt(sqrt(P(xi)) - 1) / ln (P(xi) ----------(1

More on (7):
---
tan(l*ln(P(xi))) = 0 implies that l*ln(P(xi)) = n*pi for some integer n.
ln(P(xi)) = n*pi/l
P(xi) = e^(n*pi/l) ----------(1

Above equates the Fourier polynomial for xi-th prime in terms of exponent of e with Imaginary part l of some RZF zerc

---
24g. Ramanujan Graphs, Ihara Zeta Function and Riemann Zeta Function and Special case of Complement Function
---
A graph is Ramanujan graph if it is d-regular and eigen values of its adjacency matrix are sqrt(d-1)*2 or d. Ihara ze

If a set of p-regular graphs for all primes is considered,then Riemann Zeta Function can be derived (using Ihara Zeta

[If an eigen value is t (<= q), then it can be derived that
q^s = q + (or) - sqrt(q^2 -4t) / 2
where s=a+ib. Setting a=0.5 is creating a contradiction apparently which is above divided by sqrt(q) (while equating

Above and the Informal notes in https://*sites.google.com/site/kuja27/RamanujanGraphsRiemannZetaFunctionAndIharaZetaFu*

Disclaimer: It is not in anyway an attempted proof or disproof of RH as I am not an expert in analytical number theor

(1) For some i-th prime+1 regular graph (i.e q+1 regular graph where qi is prime),

[1+qi^-s] 1
---------- = ----------
[zi(s)det(I-A*qi^-s + qi^(1-2s)*I]^(1/|V|-|E|) [1-qi^-s]

(2) Infinite product of the above terms for all prime+1 regular graphs gives the RZF in right in terms of
Ihara Zeta Function Identities product on the left - The infinite set of prime+1 regular graphs relate to
RZF zeros.

(3) For non-trivial zeros s=a+ib,RZF in RHS is zero and thus either numerator product is zero or denominator product

(4) From Handshake Lemma, it can be derived that a q-regular graph with order n(=|V| vertices) has
q*n/2 edges (=|E| edges)

(5) If numerator is set to zero in LHS,
q^(a+ib) = -1
and
cos(blogq) + isin(blogq) = -1/q^a which seems to give further contradiction

(6) If denominator is set to infinity in LHS,
[z1*z2*.....det()det()....]^1/|V|-|E| = Inf
(or)
[z1*z2*.....det()det()....]^1/|E|-|V| = 0

for some term in the infinite product of LHS which implies that
[z1*z2*.....det()det()....] = 0 which is described earlier above in the notes.

(7) More derivations of the above are in the notes uploaded in handwritten preliminary drafts at:
(7.1) https://*sites.google.com/site/kuja27/RZFAndIZF_25October2014.pdf?attredirects=0&d=1*
(7.2) http://*sourceforge.net/p/asfer/code/HEAD/tree/python-src/ComplFunction_DHF_PVsNP_Misc_Notes.pdf*

(8) By equating the determinant in (6) above to zero, written notes in (7) derive values of s=a+ib for non-Ramanujan

(9) Above doesn't look circular also because infinite product of characteristic polynomials - determinants - are inde

(10) Expression in (1) for a prime+1 regular graph can also be equated to Euler-Fourier polynomial mentioned in (24c)

[1+qi^-s] 1
---------- = ----------
[zi(s)det(I-A*qi^-s + qi^(1-2s)*I]^(1/|V|-|E|) [1-P(xi)^-s]

```
(11) qi can be replaced with Fourier polynomial P(xi),and the above becomes:
                [1-P(xi)^-2s] = [zi(s)det(I-A*P(xi)^-s + P(xi)^(1-2s)*I]^(1/|V|-|E|)
        (or)    [1-P(xi)^-2s]^(|V|-|E|) = [zi(s)det(I-A*P(xi)^-s + P(xi)^(1-2s)*I]

(12) LHS of (11) can be expanded with binomial series. Thus Euler-Fourier polynomial is coalesced into Ihara identity

(13) ACC circuits have support for mod(m) gates. Thus a trivial circuit for non-primality is set of mod(i) circuits -

(14) The Fourier polynomial of a prime P(xi) is holographic in the sense that it has information of all primes due to

(15) Without any assumption on Ihara and Riemann Zeta Functions, for any (q+1)-regular graph for prime q, just solvin

(16) In (15), even the fact that q has to be prime is redundant.  Just solving for q^s + q^(1-s) = some_eigen_value g

(17) Assuming Real(s)=0.5 from 7.1 and 7.2, it can be derived with a little more steps that eigen_value = 2*sqrt(q)*c

(18) It is not known if {Imaginary(RZF_zero)} = {Imaginary(s)}. If not equal this presents a totally different proble

(19) Maximum eigen value of (q+1)-regular graph is (q+1) and the infinite set {Imaginary(s)} can be derived from eige

(20) An experimental python function to iterate through all {Imaginary(s)} mentioned in (19) supra has been included

(21) (SOME EXPERIMENTATION ON DISTRIBUTION OF PRIMES) List of primes read by complement.py is downloaded from https:/

--------------------------------------------------
24h. PAC Learning and Complement Function Construction
--------------------------------------------------
Complement Boolean Function construction described in http://arxiv.org/abs/1106.4102 is in a way an example of PAC le
        - There are two datasets - dataset1 of size 2^n of consecutive integers and dataset2 of first n-bit prime num
        - Each element of dataset1 is mapped to i-th bit of the corresponding prime number element in the dataset2. E
        - Above step gives n probabilistic, approximate, correct learnt boolean conjunctions for each bit of all the
        - An example PAC Boolean Conjunction Learner is at: http://sourceforge.net/p/asfer/code/HEAD/tree/python-src/

------------------------------------------------------------------------------
24i. Star Complexity and Complement Function Circuit Lowerbound (related to 198)
------------------------------------------------------------------------------
Star Complexity of a Boolean Circuit is the minimum number of AND and OR gates required in monotone circuit graph. Si

----------------------------
24j. Additional references:
----------------------------
24.1 Google groups thread reference 2003 (OP done by self): https://groups.google.com/forum/#!search/ka_shrinivaasan%
24.2 Math StackExchange thread 2013: http://math.stackexchange.com/questions/293383/complement-of-a-function-f-2n-n-1
24.3 Theory of Negation - http://mentalmodels.princeton.edu/skhemlani/portfolio/negation-theory/ and a quoted excerpt
"... The principle of negative meaning: negation is a function that takes a single argument, which is a set of fully
24.4 Boolean Complementation [0 or 1 as range and domain] is a special case of Complement Function above (DeMorgan th
24.5 Interpolation - http://caig.cs.nctu.edu.tw/course/NM07S/slides/chap3_1.pdf
24.6 Formal Concept Analysis - http://en.wikipedia.org/wiki/Formal_concept_analysis, http://ijcai.org/papers11/Papers
24.7 Prime generating polynomials - http://en.wikipedia.org/wiki/Formula_for_primes
24.8 Patterns in primes - every even number > 2 is sum of 2 primes - Goldbach conjecture : http://en.wikipedia.org/wi
24.9 Arbitrarily Long Arithmetic progressions - Green-Tao theorem: http://en.wikipedia.org/wiki/Green%E2%80%93Tao_the
24.10 Prime generating functions - https://www.sonoma.edu/math/colloq/primes_sonoma_state_9_24_08.pdf
24.11 Hardness of Minimizing and Learning DNF Expressions - https://cs.nyu.edu/~khot/papers/minDNF.pdf
24.12 DNF Minimization - http://users.cms.caltech.edu/~umans/papers/BU07.pdf
24.13 DNF Minimization - http://www.cs.toronto.edu/~toni/Papers/mindnf.pdf
24.14 Riemann Zeta Function Hypothesis - all non-trivial(complex) zeroes of RZF have Re(z) = 0.5 - http://en.wikipedi
24.15 Frequent Subgraph Mining - http://research.microsoft.com/pubs/173864/icde08gsearch.pdf
24.16 Frequent Subgraph Mining - http://glaros.dtc.umn.edu/gkhome/fetch/papers/sigramDMKD05.pdf
24.17 Roots of polynomials - Beauty of Roots - http://www.math.ucr.edu/home/baez/roots/
24.18 Circuit lowerbounds (Gate Elimination, Nechiporuk, Krapchenko, etc) - http://cs.brown.edu/~jes/book/pdfs/Models
24.19 Schwartz-Zippel Lemma for Polynomial Identity Testing - http://en.wikipedia.org/wiki/Schwartz%E2%80%93Zippel_le
24.20 Schwartz-Zippel Lemma for PIT of multilinear polynomials - http://www.cs.huji.ac.il/~noam/degree.ps
24.21 Analysis of Boolean Functions - http://analysisofbooleanfunctions.org/
24.22 Riemann-Siegel Formula for computation of zeros - http://numbers.computation.free.fr/Constants/Miscellaneous/ze
24.23 RZF zeros computation - http://math.stackexchange.com/questions/134362/calculating-the-zeroes-of-the-riemann-ze
24.24 Online Encyclopedia of Integer Sequences for Prime numbers - all theorems and articles related to primes - http
24.25 Intuitive proof of Schwartz-Zippel lemma - http://rjlipton.wordpress.com/2009/11/30/the-curious-history-of-the-
24.26 Random Matrices and RZF zeros - http://www.maths.bris.ac.uk/~majpk/papers/67.pdf
24.27 Circuit for determinant - S. J. Berkowitz. On computing the determinant in small parallel time using a small nu
24.28 Mangoldt Function, Ihara Zeta Function, Ramanujan Graphs - http://lucatrevisan.wordpress.com/2014/08/18/the-rie
24.29 Multiplicity of an eigen value in k-regular graph - http://math.stackexchange.com/questions/255334/the-number-o
24.30 PAC Learning - http://www.cis.temple.edu/~giorgio/cis587/readings/pac.html
24.31 Pattern in Prime Digits - [Oliver-Kannan Soundarrajan] - http://arxiv.org/abs/1603.03720 - Prime numbers which
24.32 Pattern in Prime Digits - http://www.nature.com/news/peculiar-pattern-found-in-random-prime-numbers-1.19550 - a
24.33 Pattern in Primes - Ulam's Spiral - [Stainslaw Ulam] - https://www.alpertron.com.ar/ULAM.HTM - Prime numbers ar
24.34 Theory of Negation (Broken URL in 24.3 updated) - http://mentalmodels.princeton.edu/papers/2012negation.pdf
24.35 Prime Number Theorem - n-th prime is ~ O(nlogn)
24.36 Riemann Hypothesis prediction of prime distribution - Integral_2_p(n)_[dt/logt] = n + O(sqrt(n(logn)^3)
24.37 Brainfuck Turing Machine Compiler - https://esolangs.org/wiki/Brainfuck
24.38 Laconic Turing Machine Compiler - https://esolangs.org/wiki/Laconic
24.39 Circuit Complexity Lowerbound for Explicit Boolean Functions - http://logic.pdmi.ras.ru/~kulikov/papers/2011_3r
```

24.40 Star Complexity of Boolean Function Circuit - [Stasys Jukna] - http://www.thi.informatik.uni-frankfurt.de/~jukr
24.41 Random Matrices, RZF zeroes and Quantum Mechanics - https://www.ias.edu/ideas/2013/primes-random-matrices
24.42 Jones-Sato-Wada-Wiens Theorem - Polynomial of 25 degree-26 variables for Prime Diophantine set - http://www.mat
24.43 Energy levels of Erbium Nuclei and zeros of Riemann Zeta Function - http://seedmagazine.com/content/article/pri

##

25. (FEATURE - DONE) Approximating with some probability distribution - Gaussian, Binomial etc., that model the proba

(FEATURE - DONE) 26. Streaming algorithms - Finding Frequency moments, Heavy Hitters(most prominent items), Distinct
(FEATURE - DONE) 26.1 Implementation of LogLog and HyperLogLog Counter(cardinality - distinct elements in streamed mu
(FEATURE - DONE) 26.2 Parser and non-text file Storage framework for Realtime Streaming Data - Hive, HBase, Cassandra
(FEATURE - DONE) 26.3 Python scripts for Stock Quotes Data and Twitter tweets stream search data for a query.

References:

26.4 https://gist.github.com/debasishg/8172796

27. (FEATURE - DONE) Usual Probabilistic Measures of Mean, Median, Curve fitting on the numeric data. Python+RPy2+R i

28. (FEATURE - DONE - using python, R+rpy2) Application of Discrete Fourier Transform(using R), LOESS(using R), Linea

29. (FEATURE - DONE) Least Squares Method on datapoints y(i)s for some x(i)s such that f(x)~y needs to be found. Comp

(FEATURE - DONE)30. K-Means and kNN Clustering(if necessary and some training data is available) - unsupervised and s

31. (FEATURE - DONE) Discrete Hyperbolic Factorization - Author has been working on a factorization algorithm with el

32. (FEATURE - DONE) Multiple versions of Discrete Hyperbolic Factorization algorithms have been uploaded as drafts i
        32.1) http://sourceforge.net/projects/acadpdrafts/files/DiscreteHyperbolicFactorization_UpperboundDerivedWit
        32.2) http://sourceforge.net/projects/acadpdrafts/files/DiscreteHyperbolicPolylogarithmicSieveForIntegerFact

33. (DONE) NC PRAM version of Discrete Hyperbolic Factorization:
        33.1) An updated NC PRAM version of Discrete Hyperbolic Factorization has been uploaded at:
http://sourceforge.net/projects/acadpdrafts/files/DiscreteHyperbolicPolylogarithmicSieveForIntegerFactorization_PRAM_

        33.2) Preliminary Design notes for CRCW PRAM implementation of the above is added to repository at: http://sc

34. (FEATURE - DONE) An updated draft version of PRAM NC algorithm for Discrete Hyperbolic Factorization has been upl
        34.1 LaTeX - http://sourceforge.net/projects/acadpdrafts/files/DiscreteHyperbolicPolylogarithmicSieveForInteg

        34.2 PDF - http://sourceforge.net/projects/acadpdrafts/files/DiscreteHyperbolicPolylogarithmicSieveForIntegeI

Additional references:

        34.3) Above PRAM k-merge algorithm implies Factorization is in NC, a far more audacious claim than Factorizat

        34.4) (IMPORTANT OPTIMIZATION OVER 34.1, 34.2 ABOVE) Instead of [BerkmanSchieberVishkin] an O(logloglogN) alg

        34.5) Randomized Algorithms ,Rajeev Motwani and Prabhakar Raghavan, Chapter 12 on Distributed and Parallel Al
put size and randomized algorithms for Parallel sort - Definition 12.1 of NC (Page 336) - "NC consists of languages t
th O(logN) time and O(N^k) PRAM processors".

        34.6) Also an alternative merge algorithm in constant depth polysize circuit described in Chandra-Stockmeyer-

        34.7) [RichardKarp-VijayaRamachandran] define the inclusion of PRAM models in NC in page 29 of http://digital

        34.8) Parallel RAM survey - http://www.cs.utoronto.ca/~faith/PRAMsurvey.ps

        34.9) Related: Quantum Search using Grover Algorithm over unsorted lists can be done in O(sqrt(N)) - https://

        34.10) Recent Advances in All Nearest Smaller Values algorithms:
                34.10.1) ANSV in hypercube - Kravets and Plaxton - http://www.cs.utexas.edu/~plaxton/pubs/1996/ieee_t
                34.10.2) ANSV lowerbound - Katajainen - http://www.diku.dk/~jyrki/Paper/CATS96.ps - omega(n) processo
                34.10.3) ANSV in BSP machines - Chun Hsi Huang - http://www.cse.buffalo.edu/tech-reports/2001-06.ps

        34.11) The crucial fact in the above is not the practicality of having order of n parallel processors with RA

        34.12) Rsync - PhD thesis - chapters on external and internal sorting - https://www.samba.org/~tridge/phd_the

        34.13) Handbook of Parallel Computing - [SanguthevarRajasekaran-JohnReif] - http://www.engr.uconn.edu/~rajase

        34.14) [Eric Allender] - NC^1 and EREW PRAM - March 1990 - https://groups.google.com/forum/#!topic/comp.theor

        34.15) Efficient and Highly Parallel Computation - [JeffreyFinkelstein] - https://cs-people.bu.edu/jeffreyf/s

        34.16) There is a commercially available Parallel CRCW RAM chip implementation by NVIDIA CUDA GPUs - http://c

34.17) StackExchange thread on Consequences of Factorization in P - http://cstheory.stackexchange.com/questic

34.18) What happened to PRAM - http://blog.computationalcomplexity.org/2005/04/what-happened-to-pram.html - (

34.19) [Page 29 - HooverGreenlawRuzzo] - "... but there is no a priori upper bound on the fanout of a gate. H

34.20) (DONE-BITONIC SORT IMPLEMENTATION) In the absence of PRAM implementation, NC Bitonic Sort has been inv

34.21) An important point to note in above is that an exhaustive search in parallel would always find a facto

34.22) NC Computational Geometric algorithms - [AggarwalChazelleGuibasDunlaingYap] - https://www.cs.princetor

34.23) Parallel Computational Geometry Techniques - [MikhailAtallah] - http://docs.lib.purdue.edu/cgi/viewcor

34.24) [DexterKozen-CheeYap] - Cell Decomposition is in NC - http://anothersample.net/order/fe1d53539cff07d3e

34.25) Cell decomposition for tesselation of hyperbola can be created in two ways. In the first example, set

34.26) In another example Cell decomposition is created by intersection of integer y-axis lines and hyperbola

34.27) NVIDIA CUDA Parallel Bitonic Sort Implementation Reference - http://developer.download.nvidia.com/comp

34.28) Tiling of Hyperbolic curve: The discretization step of a hyperbola to form set of contiguous tiled seg

34.29) The tiling of hyperbolic curve is done by deltax = N/[y(y+1)] which is a process similar to low-pas

34.30) Above algorithm ignores Communication Complexity across PRAMs or Comparator nodes on a Cloud.

34.31) Comparison of Parallel Sorting Algorithms (Bitonic, Parallel QuickSort on NVIDIA CUDA GPU etc.,) - htt

34.32) NC-PRAM Equivalence and Parallel Algorithms - [David Eppstein] - https://www.ics.uci.edu/~eppstein/pub

34.33) Parallel Merge Sort - [Richard Cole - https://www.cs.nyu.edu/cole/] - best known theoretical parallel

34.34) NC-PRAM equivalence - http://courses.csail.mit.edu/6.854/06/notes/n32-parallel.pdf

34.35) Theorem 13 - PRAM(polylog,log) = uniform-NC - www.wisdom.weizmann.ac.il/~oded/PS/CC/l27.ps - This ment

34.36) PRAM Implementation Techniques - http://www.doc.ic.ac.uk/~nd/surprise_95/journal/vol4/fcw/report.html

34.37) Logarithm Time Cost Parallel Sorting - [Lasse Natvig] - Survey on Batcher's Bitonic Sort, AKS Sorting

34.38) What is wrong with PRAM - "Too much importance is placed on NC . In particular, algorithms which have

34.39) NC-PRAM equivalence - https://www.ida.liu.se/~chrke55/courses/APP/ps/f2pram-2x2.pdf - "set of problems

34.40) Number of PRAMs in NC definition - http://cs.stackexchange.com/questions/39721/given-a-pram-may-use-ar

34.41) EURO-PAR 1995 - Input size in Parallel RAM algorithms - https://books.google.co.in/books?id=pVpjOwUHEa

34.42) Batcher's Bitonic Sort is in NC - https://web.cs.dal.ca/~arc/teaching/CS4125/Lectures/03b-ParallelAnal

34.43) Brief Overview of Parallel Algorithms, Work-Time Efficiency and NC - [Blelloch] - http://www.cs.cmu.ed

34.44) Sorting n integers is in NC - [BussCookGuptaRamachandran] - www.math.ucsd.edu/~sbuss/ResearchWeb/Boole

34.45) PRAM and Circuit Equivalence - [Savage] - http://cs.brown.edu/~jes/book/pdfs/ModelsOfComputation.pdf -

34.46) Efficient Parallel Computation = NC - [AroraBarak] - http://theory.cs.princeton.edu/complexity/book.pd

34.47) Parallel sorting and NC - http://courses.csail.mit.edu/6.854/06/notes/n32-parallel.pdf

34.48) Parallel sorting in PRAM model - http://www.math.ias.edu/~avi/PUBLICATIONS/MYPAPERS/MW87/MW87.pdf

34.49) Parallel Sorting is in NC - http://www.toves.org/books/distalg/distalg.pdf - Section 5 (Page 17)

34.50) Parallel K-Merge Algorithm for merging k sorted tiles into a single sorted list - LazyMerge - www.cs.n

34.51) Timsort and its parallel implementation in Java, Python and Android for parallel merge sort of arrays

34.52) Comparison of Parallel and Sequential Sorts - https://github.com/darkobozidar/sequential-vs-parallel-s

34.53) Trigonometric Functions over Finite Galois Fields (of prime power order) - https://arxiv.org/pdf/1501.

34.54) Adaptive Bitonic Sorting - http://epubs.siam.org/doi/abs/10.1137/0218014 - very old paper (1989) on PR

34.55) Efficient Parallel Sorting - improvement of Cole's Parallel Sort and AKS sorting networks: sorting net

34.56) Computational Pixelation Geometry Theorems - https://arxiv.org/pdf/1105.2831v1.pdf - defines planar pi

34.57) Approximating a function graph by planar pixelation - http://www3.nd.edu/~lnicolae/Pixelations-beam.pd

34.58) Resource Oblivious Sorting on Multicores - [RichardCole-VijayaRamachandran] - https://arxiv.org/pdf/15

34.59) NC and various categories of PRAMs - https://pdfs.semanticscholar.org/5354/99cf0c2faaaa6df69529605c87b

34.60) PRAM-NC equivalence, definition of input size - ftp://ftp.cs.utexas.edu/pub/techreports/tr85-17.pdf -

34.61) PRAM model and precise definition of input size in PRAM models - https://www.cs.fsu.edu/~engelen/cours

34.62) Simulation of Parallel RAMs by Circuits - http://www.cstheory.com/stockmeyer@sbcglobal.net/sv.pdf

34.63) PRAM Models and input size - [Joseph JaJa] - https://www.cs.utah.edu/~hari/teaching/bigdata/book92-Ja

34.64) Input Size for PRAMs, NC-PRAM equivalence, Cook-Pippenger Thesis, Brent's Speedup Lemma - www.csl.mtu.

34.65) Pixelation of Polygons - On Guarding Orthogonal Polygons with Bounded Treewidth - [Therese Biedl and S

34.66) A Comparison of Parallel Sorting Algorithms on Different Architectures- [NANCY M. AMATO, RAVISHANKAR I

34.67) Bitonic Sort Implementation - [Amrutha Mullapudi] - https://www.cse.buffalo.edu//faculty/miller/Course

34.68) PRAMs and Multicore architectures - http://blog.computationalcomplexity.org/2008/04/revenge-of-paralle

34.69) NC and PRAMs - http://www.cse.iitd.ernet.in/~subodh/courses/CSL860/slides/3pram.pdf

34.70) Geometric Searching, Ray Shooting Queries - [Michael T.Goodrich] - http://www.ics.uci.edu/~goodrich/pu

34.71) Planar Point Location in Parallel - ftp://ftp.cs.brown.edu/pub/techreports/89/cs89-45a.pdf - Section 4

34.72) One dimensional Point Location - https://www.csun.edu/~ctoth/Handbook/chap38.pdf - Section 38.1 - List

34.73) Hoover-Klawe-Pippenger Algorithm for converting arbitrary circuit to bounded fanout 2 - [Ogihara-Anime

34.74) PRAM definition of NC - [Kristoffer Arnsfelt Hansen] - https://users-cs.au.dk/arnsfelt/CT08/scribenote

34.75) Factorization, Planar Point Location in parallel, Cascading, Polygon pixelation of hyperbola - [Mikhai

34.76) Multicores,PRAMs,BSP and NC - [Vijaya Ramachandran] - https://womenintheory.files.wordpress.com/2012/6

34.77) Word Size in a PRAM - https://hal.inria.fr/inria-00072448/document - Equation 5 - logN <= w (N is inpu

34.78) Prefix Sum Computation and Parallel Integer Sorting - [Sanguthevar Rajasekaran, Sandeep Sen] - http://

35. (THEORY) Above Discrete Hyperbolic Factorization can be used as a numerical dataset analysis technique. Discrete

##
B. EXPERIMENTAL **NON**-STATISTICAL INFERENCE MODEL BASED ON ALREADY KNOWN THEORETICAL COMPUTER SCIENCE RESULTS:
##

36.1. (FEATURE - DONE-WordNet Visualizer implementation for point 15) The classification using Interview Algorithm De

36.2. (THEORY) Thus if number of classes and documents are infinite, an infinite hypergraph is obtained. This is alsc

37.(FEATURE - PoC WordNet Visualizer Closure DONE) Above multiplanar hypergraph is quite related as a theoretical cor

38.(THEORY) But above is not context free because context is easily obtainable from the hyperedge which connects mult

39.(THEORY) An interesting academic question to pose is - "Does the above hypergraph have a strong connectivity and a

40.(THEORY) A recent result of Rubik's cube (http://www.cube20.org/) permutations to get solution having a constant u

41.(THEORY) Mapping Rubik's Cube to Concept Wide Hypergraph above - Each configuration in Rubik's Cube transition fur

42.(THEORY) The Multiplanar Primordial Field Turing Machine model described by the author in "Theory of Shell Turing

43.(THEORY) Instead of a wordnet if a relationship amongst logical statements (First Order Logic etc.,) by logical im

44.(THEORY) Implication graphs of logical statements can be modelled as a Random Growth Network where a graph is ran

45.(THEORY) If a statistical constraint on the degree of this graph is needed, power law distributions (Zipf, Paretc

46.(THEORY) Set cover or Hitting Set of Hypergraph is a Transversal of hypergraph which is a subset of set of vertice

47.(THEORY) Random walk on the above concept hypergraph, Cover time etc., which is a markov process. Probably, Cover

48.(THEORY - IMPLEMENTATION - DONE) Tree decomposition or Junction Trees of concept hypergraph with bounded tree wid

49.(THEORY) 2(or more)-dimensional random walk model for thinking process and reasoning (Soccer model in 2-dimension

---

As an experimental extension of point 47 for human reasoning process, random walk on non-planar Concept hypergraph (

```
1111
1112  ##############################################################################
1113  C.      Slightly Philosophical - Inferences from conflicting opinions:
1114  ##############################################################################
1115
1116  50. (DONE) As an example, if there are "likes" and "dislikes" on an entity which could be anything under universe - h
1117
1118  51. (DONE) Another philosophical question is what happens to the majority voting when every voter or witness is wrong
1119
1120  52. (THEORY) There is a realword example of the above - An entity having critical acclaim does not have majority accl
1121
1122  53. (THEORY) Items 50,51,52 present the inherent difficulty of perfect judgement or perfect inference. On a related n
1123
1124  ------------------------------------------------------------------------------------------------------------------------
1125  53.1 (THEORY) Judge Boolean Function (there doesn't seem to be an equivalent to this in literature - hence it is defi
1126  ------------------------------------------------------------------------------------------------------------------------
1127  A Judge Boolean function f:(0,1)^n->(0,1) has two classes of input sets of bit sets (x1,x2,x3,...) and (y1,y2,y3,...)
1128          - LHS PRG/Dictator boolean function choice can also be thought of as randomly chosen Judge boolean function (
1129          - In RHS, each voter has a judge boolean function above and NoiseStability of Boolean Function Composition of
1130          - An example: From https://sites.google.com/site/kuja27/InterviewAlgorithmInPSPACE.pdf?attredirects=0, Judge
1131          - QBF is PSPACE-complete and hence Judge Boolean Function is PSPACE-complete.
1132          - Thus LHS of P(good) is in PSPACE (a pseudorandomly chosen Judge Boolean Function) while RHS of P(good) is s
1133          - Since Judgement or Choice among two adversaries is a non-trivial problem, it is better to have each Voter's
1134          - Reference: QBF-SAT solvers and connections to Circuit Lower Bounds - [RahulSanthanam-RyanWilliams] - http:/
1135          - A complex Judging scenario: Doctrinal Paradox (or) Discursive dilemma - Paradoxes in Majority voting where
1136
1137  ------------------------------------------------------------------------------------------------------------------------
1138  53.2 (THEORY) P(Good) Convergence, Voter TQBF function, PH-completeness and EXP-completeness - Adversarial reduction
1139  ------------------------------------------------------------------------------------------------------------------------
1140  P(Good) RHS is in PH if depth restricted and in EXP if depth unrestricted. LHS is a pseudorandomly chosen Judge Boole
1141  Each Voter can be thought of as simulator of both adversaries in Chess i.e Voter TQBF has two sets of inputs x1,x2,x3
1142
1143  A special case of Complexity of Quantified 3-CNF Boolean Forumlas is analyzed in http://eccc.hpi-web.de/report/2012/0
1144
1145  Depth Hierarchy Theorem ,a recent result in 2015 in http://arxiv.org/pdf/1504.03398.pdf proved PH to be infinite rela
1146
1147  ------------------------------------------------------------------------------------------------------------------------
1148  53.3 (THEORY- *IMPORTANT* ) Conditions for complexity classes of LHS and RHS of P(Good) summation being equated
1149  ------------------------------------------------------------------------------------------------------------------------
1150  RHS of P(Good) is either PH-complete or EXP-complete as mentioned in 53.2.
1151  LHS of P(Good) is a pseudorandomly chosen Judge Boolean Function while RHS is a huge mixture of judge boolean functio
1152
1153  Probability of LHS Judge Boolean Function belonging to a complexity class C with NoiseStability p is =
1154  Probability of LHS PRG choice Judge Boolean Function in complexity class C * Probability of Goodness in terms of Nois
1155   c/n * p where c is the number of judge boolean functions in complexity class C and n is the total number of judge bo
1156
1157  When p=1 and c=n LHS is 1 and is a complexity class C algorithm to RHS PH-complete or EXP-complete DC circuit when RH
1158    - boolean function composition in RHS is in some fixed circuit complexity class (PH-complete or EXP-complete) and c
1159    - LHS has to have 100% noise stability
1160    - RHS has to have 100% noise stability
1161  Thus above conditions are to be satisified for equating complexity classes in LHS and RHS to get a lowerbound result.
1162
1163  If LHS and RHS of P(Good) are not equatable, probability of goodness of LHS and RHS can vary independently and have a
1164          cardinality(RHS) + cardinality(LHS) = cardinality(Universal set of judge boolean functions) (or)
1165          cardinality(RHS) + 1 = cardinality(Universal set)
1166  which is the proverbial "one and all".
1167
1168  Assuming there exists a boolean percolation function with noise stability 100%, voter with such a boolean function co
1169
1170  ------------------------------------------------------------------------------------------
1171  53.4 (THEORY) Variant 1 of Percolation as a Judge boolean function (related to 14 above)
1172  ------------------------------------------------------------------------------------------
1173  Percolation boolean function returns 1 if there is a left-right crossing event in a path within the Z^2 grid random g
1174  There are n parameters on which the voter or a judge weighs two adversaries a and b (each take the 2 coordinates in t
1175
1176  ------------------------------------------------------------------------------------------
1177  53.5 (THEORY) Variant 2 of Percolation as a Judge boolean function (related to 14 above)
1178  ------------------------------------------------------------------------------------------
1179  53.5.1 Both x and y axes have the criteria common to both adversaries a and b as coordinates.
1180  53.5.2 If adversary a is equal to b for a criterion c, then there is a horizontal left-right edge with probability p.
1181  53.5.3 If adversary b is better than a for criterion c, then there is a vertical edge (down-top) with probability p.
1182  53.5.4 If adversary a is better than b for criterion c, then there is a vertical edge (top-down) with probability p.
1183  -----------------
1184  53.5.5 Example 1:
1185  -----------------
1186  criterion    a    b
1187  c1           5    4 (top-down)
1188  c2           3    5 (down-top)
1189  c3           2    2 (left-right)
1190  c4           4    6 (down-top)
1191  c5           7    8 (down-top)
```

```
        --------------------
            21    25  (a < b)
        --------------------
---------------
53.5.6 Example 2:
---------------
criterion    a    b
c1           2    1 (top-down)
c2           7    2 (top-down)
c3           5    5 (left-right)
c4           3    8 (down-top)
c5           9    7 (top-down)
        --------------------
            26    23  (a > b)
        --------------------
53.5.7 Length of each edge is proportional to the difference between score(a) and score(b) at criterion c.
53.5.8 Lemma: a is better than b implies a top-down crossing event and b is better than a implies down-top crossing e
Proof: If a is better than b in majority of the criteria then the path is more "top-down"-ish due to 53.5.4 (top-left
53.5.9 Above variant of percolation boolean function returns 1 if path is top-down (a > b) else 0 if path is down-top
53.5.10 Previous description is an equivalence reduction of percolation and judging two adversaries.
53.5.11 To differentiate top-down and down-top left-right paths, comparator circuits (constant depth, polysize) and n

-----------------------------------------------------------------------------------------------------------
53.6 (THEORY) Circuit for Percolation as a Judge Boolean Function
-----------------------------------------------------------------------------------------------------------
For an m * m grid with random edges, maximum length of the left-right path is m^2 (traverses all grid cells). Each bo
  - 1 if both x-axis coordinate of the leftmost ordered pair is 0 and x-axis coordinate of the rightmost ordered pair
  - else 0
which requires comparators. Sorting networks of depth O((logn)^2*n) and size O(n(logn)^2) exist for algorithms like E
  - Sorting network that sorts the coordinates left-right in NC1 or NC2
  - Comparator circuits in constant depth and polynomial size
which together are in NC. This is non-uniform NC with advice strings as the circuit depends on percolation grid - e.g

Above percolation circuit based on Sorting networks is not optimal for arbitrarily large inputs. As per Noise Stabili

-----------------------------------------------------------------------------------------------------------
53.7 (THEORY) P/Poly and P(Good) LHS and RHS - Karp-Lipton and Meyer's Theorems
-----------------------------------------------------------------------------------------------------------
P/poly is the circuit class of polynomial size with polynomial advice strings and unrestricted depth. Thus P/poly is

If RHS is simply NP (if depth restricted and unlikely for arbitrary number of electorate) then LHS is a P/poly algori

EXPTIME version of this is Meyer's Theorem which states that if EXPTIME is in P/poly (which is the hardest case for R

References:
----------
53.7.1 Relation between NC and P/poly - http://cs.stackexchange.com/questions/41344/what-is-the-relation-between-nc-a
53.7.2 P/poly - https://en.wikipedia.org/wiki/P/poly
53.7.3 Karp-Lipton Theorem - https://en.wikipedia.org/wiki/Karp%E2%80%93Lipton_theorem
53.7.4 NP in P/poly implies AM=MA - [Arvind, Vikraman; K��bler, Johannes; Sch��ning, Uwe; Schuler, Rainer (1995),] -
53.7.5 BPP,PH,P/poly,Meyer's theorem etc., - http://www.cs.au.dk/~arnsfelt/CT10/scribenotes/lecture9.pdf
53.7.6 Descriptive Complexity - https://books.google.co.in/books?id=kWSZ0OWnupkC&pg=PA177&lpg=PA177&dq=NC1/poly&sourc
53.7.7 Non-uniform computation - https://courses.engr.illinois.edu/cs579/sp2009/slides/cc-s09-lect10.pdf

-----------------------------------------------------------------------------------------------------------
53.8 (THEORY) An example Majority+SAT majority voting circuit, P(Good) convergence and some observations on Noise Sta
-----------------------------------------------------------------------------------------------------------
Let the number of voters be 8 (=2^3). Each voter has a boolean function of 3 variables - x1,x2 and x3. From Boolean F

If the RHS doesn't converge to 1 and thus is in BPP and Noise Stability of LHS is 100% (e.g by PRG choice of a percol

If the RHS has a perfect k-SAT boolean decision function for all voters, size of the circuit is super-polynomial and

If LHS of P(Good) is in NC/log (Percolation as Judge boolean function) and RHS doesn't converge to 100% (i.e Majority

Subtlety 1: For single voter in RHS with a 3-SAT voter judge boolean function, RHS is NP-complete from voting complex

Subtlety 2: [Impagliazzo-Wigderson] theorem states that if there exists a decision problem with O(exp(n)) runtime and

NC/log circuits can be made into a uniform circuits by hardwiring advice strings which are logarithmic size. There ar

References:
----------
53.8.1 Pseudorandomness - [Impagliazzo-Wigderson] - http://people.seas.harvard.edu/~salil/pseudorandomness/pseudoranc
53.8.2 XOR Lemma [Yao] and [Impagliazzo-Wigderson] theorem - theory.stanford.edu/~trevisan/pubs/stv99stoc.ps

-----------------------------------------------------------------------------------------------------------
53.9 (THEORY) Possible Scenarios in P(Good) LHS and RHS summation
-----------------------------------------------------------------------------------------------------------
53.9.1 LHS = RHS is 100%                -   LHS and RHS are equally efficient (and can be equated to get a lowerbound
53.9.2 LHS is < 100% and RHS is < 100%  -   RHS is more efficient than LHS and both sides are in BP* classes
```

```
and LHS < RHS
53.9.3 LHS is < 100% and RHS is < 100%  -   LHS and RHS are equally efficient (and can be equated to get a lowerbound
and LHS = RHS
53.9.4 LHS is < 100% and RHS is < 100%  -   LHS is more efficient than RHS and both sides are in BP* classes
and LHS > RHS
53.9.5 LHS is 100% and RHS is < 100%    -   LHS is more efficient than RHS. LHS is non-BP algorithm to RHS BP* algori
53.9.6 LHS is < 100% and RHS is 100%    -   RHS is more efficient than LHS. RHS is non-BP algorithm to LHS BP* algori

-----------------------------------------------------------------------------------------------------------------------
53.10 (THEORY) Dictatorship, k-juntas testing and P(Good) circuit - simulation
-----------------------------------------------------------------------------------------------------------------------
LHS of P(Good) is a pseudorandom choice judge boolean function. It has to be tested for dictatorship and k-junta prop
the hardness of LHS. What is the probability of LHS pseudorandom choice being a dictator or k-junta boolean function?
        - PRG Choice outputs a Judge boolean function candidate for LHS.
        - Dictatorship or k-juntas testing algorithm is executed for this candidate PRG choice.
Expected number of steps required for choosing a dictator via PRG is arrived at by Bernoulli trial. Repeated PRG choi
p(t) = (1-k/n)^(t-1)(k/n) = (n-k)^(t-1) * k/n^t

If t is the random variable, expectation of t is:
E(t) = sigma(t*p(t)) = sigma((n-k)^(t-1)*tk/n^t)

For a property testing algorithm running in O(q), time required in finding a dictator or k-juntas is O(q*sigma((n-k)^

For k=0:
        E(t) = 0
For k=1:
        E(t) = sigma((n-1)^(t-1) * t/n^t) = 1/n + (n-1)*2/n^2 + (n-1)^2*3/n^3 + ...
             < 1/n + 2n/n^2 + 3n^2/n^3 + ...
             = 1+2+3+...+n/n
             = n(n+1)/2n
             = (n+1)/2
Thus to find one dictator approximately (n+1)/2 PRG expected choices are required. Above is not necessarily an exact

[Axiomatically k shouldn't be greater than 1 i.e there shouldn't be more than one dictator or juntas. Philosophical p

References:
-----------
53.10.1 Dictatorship,k-juntas testing - [Oded Goldreich] - http://www.wisdom.weizmann.ac.il/~oded/PDF/pt-junta.pdf

-----------------------------------------------------------------------------------------------------------------------
(THEORY-*IMPORTANT*) 53.11 Hastad-Linial-Mansour-Nisan (HLMN) Theorem and Noise Stable boolean functions - Denoisific
-----------------------------------------------------------------------------------------------------------------------
Hastad-Linial-Mansour-Nisan Theorem states that for a boolean function f:{0,1}^n->{0,1} computed by circuit of size <
fourier coefficients for large S are epsilon-concentrated:
        sigma(fourier_coeff(S)^2) < epsilon for |S| > t where t = O((log s/epsilon)^(1/d-1)).
Above implies that:
        t = O(log s/epsilon)^(d-1) = k*(log s/epsilon)^(d-1) for some constant k.
        (t/k)^(1/d-1) = log(s/epsilon)
        s = epsilon*2^((t/k)^1/(d-1))
For a noisy boolean function f with rho probability of flip, the noise operator T is applied to the Fourier expansion
        T(f) = sigma(rho^|S|*fourier_coeff(S)*parity(S)) for S in [n] and each Fourier coefficient of T(f) = rho^|S|*
From HLMN theorem, applying noise operator above (this needs to be verified) to noisy boolean function circuit Fourie
        if sigma([rho^|S|*fourier_coeff(S)]^2) <= epsilon, |S| > t where t=O((log s/epsilon)^(d-1)),
        size s = sigma([rho^|S|*fourier_coeff(S)]^2)*2^[(t/k)^1/(d-1)] substituting for epsilon and |S| > t.
        Thus size of the circuit exponentially depends on t - size of parity sets S in [n] - |S| - which can have max

If f is 100% noise stable, from Plancherel theorem:
        sigma(rho^|S|*f(S)^2) = 1, S in [n]
Stability can be written as:
        sigma_S_<_t(rho^|S|*f(S)^2) + sigma_S_>=_t(rho^|S|*f(S)^2) = 1
        [1 - sigma_S_<=_t(rho^|S|*f(S)^2)] = sigma_S_>_t(rho^|S|*f(S)^2)
Since rho < 1:
        epsilon = sigma_S_>_t_((rho^|S|*f(S))^2) < epsilon_stable = sigma_S_>_t_(rho^|S|*f(S)^2) , |S| > t
        epsilon_stable can also be written as = sigma_S_<_t_(1-rho^|S|*f(S)^2), |S| < t

        s = sigma[f(S)^2] * 2^(t/k)^1/(d-1)) , t=O((log s/epsilon)^(d-1)), |S| > t
        s_stable = [1-sigma_S_<_t(rho^|S|*f(S)^2)]*2^((t/k)^1/(d-1))  , t=O((log s_stable/epsilon_stable)^(d-1)), |S|
        (or)
        s_stable = sigma(rho^|S|*[fourier_coeff(S)]^2)*2^[(t/k)^1/(d-1)] , t=O((log s_stable/epsilon_stable)^(d-1)),
s_stable is the size of the 100% noise stable circuit for f.
From above, t with noise operator is:
        t(noise_operator)=O(log s/sigma(rho^|S|*f(S)^2)), |S| > t (or)
        t(noise_operator)=O(log s/(1-sigma_S_<_t(rho^|S|*f(S)^2))), |S| < t
which is substituted for t in s_stable above and t without noise operator is:
        t=O(log s/sigma(f(S)^2)), |S| > t
which implies that t(noise_operator) > t since denominator summation of fourier coefficients is attenuated by rho^|S|

*Implication 1 of above: If RHS is in BPP (e.g a 3-SAT voter judge boolean function circuit with < 100% noise stabili

*Implication 2 of above: If LHS of P(Good) has BP.(NC/log) circuit due to <100% noise stability, with polynomial size
```

```
Points 53.4, 53.5, 53.6 and 53.7 describe P/poly, NC/poly and NC/log circuits for Percolation Boolean Functions subje

Noise sensitive NC/log percolation circuit in LHS of P(Good) is in BP(NC/log) which requires denoisification and dera

-----------------------------------------------------------------------------------------------------------------
(THEORY-*IMPORTANT*) 53.12 Hastad-Linial-Mansour-Nisan (HLMN) Theorem and Noise Stable boolean functions - Denoisific
-----------------------------------------------------------------------------------------------------------------
Above version derived by rewriting HLMN in terms of s(size) and t(size of fourier basis) is not too appealing. Basica
        For |S| > t, Summation(fourier_coeff(S)^2) <= 2*(Size)*2^(-t^(1/d)/20)
For Noise-operated Fourier Spectra, the summation is augmented with rho(noise operator):
        For |S| > t, Summation(rho^|S|*fourier_coeff(S)^2) <= 2*(Size)*2^(-t^(1/d)/20)
Applying Plancherel theorem and stability=1:
    ---------------------------------------------------------------------------------------------------------
    |   For |S| < t, 1-Summation(rho*fourier_coeff(S)^2) <= 2*(Size)*2^(-t^(1/d)/20)
    |   => [1-Summation(rho^|S|*fourier_coeff(S)^2)] * 2^((t^(1/d)/20)-1) <= Size
    ---------------------------------------------------------------------------------------------------------
which is the circuit size lowerbound in terms of noise probability and fourier coefficients. As rho increases from 0
size decreases, that is, low noise requires high circuit size. Partial Derivative of size bound wrt rho:
    -Summation(fourier_coeff(S)^2)] * 2^((t^(1/d)/20)-1) <= doe(Size)/doe(rho)
which points to exponential - in t and d - decrease in size (because t=O(n) and d=O(function(n)) with linear increase
        ---------------------------------------------------------------------------------------------
        | P(Good) binomial summation in its circuit version can not have polynomial size circuits     |
        | when LHS and RHS converge with 100% noise stability.                                        |
        ---------------------------------------------------------------------------------------------

For example, if LHS is a percolation logdepth circuit and RHS is single voter with 3-SAT NP-complete circuit, when LH

        LHS circuit size >= O(2^n^(1/polylog(n))) = superpolynomial, quasiexponential lowerbound, depth is polylog(n)
implying LHS percolation circuit could become superpolynomial sized when noise stability is 1 and not in NC despite b

        RHS circuit size >= O(2^n^(1/d)) = higher superpolynomial, quasiexponential lowerbound, depth is arbitrary ,c
and thus LHS and RHS circuits are equally efficient algorithms with differing circuit sizes. As RHS is NP-complete, L
[Open question: Does this imply superpolynomial size lowerbounds for NP and thus P != NP]


Reference:
-----------
53.12.1 Derandomizing BPNC - [Periklis Papakonstantinou] - http://iiis.tsinghua.edu.cn/~papakons/pdfs/phd_thesis.pdf
53.12.2 Derandomization basics - [Salil] - Enumeration - http://people.seas.harvard.edu/~salil/pseudorandomness/basic
53.12.3 Simplified Hitting Sets Derandomization of BPP - [Goldreich-Vadhan-Wigderson] - http://www.wisdom.weizmann.ac
53.12.4 Derandomizing BPNC - Existence of Hitting Set Generators for BPNC=NC - [Alexander E.Andreev, Andrea E.F.Cleme
53.12.5 Linial-Mansour-Nisan Theorem - Lemma 7 (Main Lemma) - http://www.ma.huji.ac.il/~ehudf/courses/anal09/LMN.pdf:
        For |A| > t, Summation(fourier_coeff(A)^2) <= 2*(Size)*2^(-t^(1/d)/20)
53.12.6 The Computational Complexity Column [Allender-Clementi-Rolim-Trevisan] - theory.stanford.edu/~trevisan/pubs/e

-----------------------------------------------------------------------------------------------------------------
53.13. (THEORY) Partial Derivative of Circuit size wrt rho (noise probability) - increase in circuit size for Denoisi
-----------------------------------------------------------------------------------------------------------------
From 53.11 above, size of denoisified circuit from HMLN theorem, and Stability in terms of Fourier coefficients with
        s_stable = [1-sigma_S_<_t(rho^|S|*f(S)^2)]*2^((t/k)^1/(d-1))  , t=O((log s_stable/epsilon_stable)^(d-1)), |S|
        (or)
        s_stable = sigma(rho^|S|*[fourier_coeff(S)]^2)*2^[(t/k)^1/(d-1)] , t=O((log s_stable/epsilon_stable)^(d-1)),
Lower bound of exponent increases as sigma(rho^|S|*f(S)^2) decreases in denominator of exponent subject to stability
        [1 - sigma_S_<=_t(rho^|S|*f(S)^2)] = sigma_S_>_t(rho^|S|*f(S)^2)
which implies that sigma_S>t_(rho^|S|*f(S)^2) decreases when sigma_S<t_(rho^|S|*f(S)^2) increases. This is Consequenc

Partial derivative of Circuit size wrt to rho can be derived as:
        delta_s/delta_rho = sigma_S>t_(|s|*rho^(|s|-1) * f(S)^2) / (1 - sigma_S<t_(rho^|S|*f(S)^2)) where delta_s is

-----------------------------------------------------------------------------------------------------------
53.14. (THEORY) Hastad-Linial-Mansour-Nisan Theorem, Circuit size, Noise Stability and BP* classes
-----------------------------------------------------------------------------------------------------------
Percolation circuit defined previously has:
- Sorting Network for creating path sequentially on grid (NC logdepth and polysize)
- Comparators (Constant depth and polysize) are required for comparing the leftmost and rightmost coordinates.
Together Percolation is in non-uniform NC. This circuit depends on number of points on the left-right path which is p
Therefore percolation is in NC/poly because size of list of advice strings could be O((n^2)log(n^2)) and it may not h

From Linial-Mansour-Nisan Theorem - Lemma 7 (Main Lemma) - http://www.ma.huji.ac.il/~ehudf/courses/anal09/LMN.pdf:
        For |A| > t, Summation(fourier_coeff(A)^2) <= 2*(Size)*2^(-t^(1/d)/20)
With noise operator, noise stability=1 from Plancherel's theorem:
    ---------------------------------------------------------------------------------------------------------
    |   For |S| < t, 1-Summation(rho^|S|*fourier_coeff(S)^2) <= 2*(Size)*2^(-t^(1/d)/20)
    |   Corollary: For |S| > t, Summation(rho^|S|*fourier_coeff(S)^2) <= 2*(Size)*2^(-t^(1/d)/20)
    |   => [1-Summation(rho^|S|*fourier_coeff(S)^2)] * 2^((t^(1/d)/20)-1) <= Size
    ---------------------------------------------------------------------------------------------------------
When denoisified, rho=0 and lowerbound for size:
        [1-(0^0*f(phi)^2+0+0+..)]*2^((t^(1/d)/20)-1) <= size
        [1-f(phi)^2]*2^((t^(1/d)/20)-1) <= size (as all other coefficients vanish)
when t=n, the size is lowerbounded by :
        [1-f(phi)^2]*2^((n^(1/d)/20)-1)
```

which is superpolynomial. This implies an NC/poly polysize circuit when denoisified becomes O(2^((n^(1/polylog(n)/20)

From the table that describes relation between Noise and Error(BP* and PP) in 14 previously, it is evident that Noise
if following is drawn as a Venn Diagram i.e. Noise stability solves just a fraction of total error, and remaining rec
In other words, Noisy circuit could be error-free and Noise-free circuit could have error.

x	f(x) = f(x/e)	f(x) != f(x/e) Noise
x in L, x/e in L	No error	Error
x in L, x/e not in L	Error	No error if f(x)=1,f(x/e)=0 else Error
x not in L, x/e in L	Error	No error if f(x)=0,f(x/e)=1 else Error
x not in L, x/e not in L	No error	Error

Noise percolation circuit in P(Good) summation converges in noise stability regime:
lt Pr[f(w(n)) != f(w(n)/e)] = 0, e=1/(alpha4*n^2)
n -> inf
Infinite f(n) is representable as Non-uniform NC/poly circuit though undecidable because turing machine computing the

For a 3-CNFSAT boolean function with noise, size of circuit is lowerbounded as:
[1-Summation(rho^|S|*fourier_coeff(S)^2)] * 2^((t^(1/d)/20)-1) <= Size , for constant depth d.
For rho=1,
[1-Summation(fourier_coeff(S)^2)] * 2^((t^(1/d)/20)-1) <= Size , for constant depth d, |S| < t.

Gist of the above:
53.14.1 Size of percolation logdepth circuit (with noise) rho=1:
(f([n])^2) * 2^(n^(1/polylog(n-1)/20)-1) <= size
53.14.2 Size of depth d circuit (with noise) rho=1:
(f([n])^2) * 2^((n-1)^(1/d/20)-1) <= size
53.14.3 Size of percolation logdepth circuit (with noise) rho=0:
(1-f([phi])^2) * 2^(n^(1/polylog(n)/20)-1) <= size, phi is empty set
53.14.4 Size of depth d circuit (with noise) rho=0:
(1-f([phi])^2) * 2^(n^(1/d/20)-1) <= size, phi is empty set

53.15 (THEORY) Special Case of HLMN theorem for PARITY as 3-SAT and counterexample for polynomial circuit size

---

From HMLN theorem:
Summation(fourier_coeff(S)^2) * 2^((t^(1/d)/20)-1) <= Size , for constant depth d, |S| > t.
When t=n-1, fourier series has only coefficient for the parity set that has all variables:
fourier_coeff([n])^2 * 2^(((n-1)^(1/d)/20)-1) <= size, for constant depth d.
For a circuit of size polynomial (n^k) and depth 3:
summation(fourier_coeff([n])^2) <= n^k / 2^((t^(1/3)/20)-1).
and for t=n-1:
fourier_coeff([n])^2 <= n^k / 2^(((n-1)^(1/3)/20)-1).
From definition of fourier coefficients:
fourier_coeff([n]) = summation(f(x)*(-1)^(parity(xi)))/2^n for all x in {0,1}^n
Maximum of fourier_coeff([n]) occurs when f(x)=1 for (-1)^parity(xi) = 1 and when f(x)=-1 for (-1)^parity(xi) = -1 to
Maximum(fourier_coeff([n])) = 2^n/2^n = 1 (this implies all other fourier coefficients are zero)
Which happens when f(x) is equivalent to Parity function.
An example PARITY as 3-CNFSAT from DIMACS32 XORSAT [Jing Chao Chen - http://arxiv.org/abs/cs/0703006] that computes p
(x1 V not x2 V not x3) /\ (not x1 V x2 V not x3) /\ (not x1 V not x2 V x3) /\ (x1 V x2 V x3)
Previous PARITY SAT accepts odd parity strings. Complementing variables in each clause makes it accept even parity:
(not x1 V x2 V x3) /\ (x1 V not x2 V x3) /\ (x1 V x2 V not x3) /\ (not x1 V not x2 V not x3)
Assumption: PARITY as 3-CNFSAT has polynomial size.
PARITY 3-SAT is satisified by odd parity bit strings and is NP-complete because this is search version of parity - it
=> 1^2 <= n^k / 2^(((n-1)^(1/3)/20)-1) which is a contradiction to assumption (denominator is exponential) that P

From above, for some circuit size of PARITY as 3-CNFSAT, there exists t(=O(n)) such that:
2^(((t)^(1/3)/20)-1) <= size / summation(f(S)^2) , |S| > t
2^(((t)^(1/3)/20)-1) <= size (for all t < n-1, fourier coefficients of PARITY 3-SAT are 0, and this bound is m
i.e for all values of t, circuit size is bounded from below by a quantity exponential in function of n (which could b

(How can a special case of NP-complete problem have superpolynomial size circuits? Is this a counterexample and thus

References:

53.15.1 Circuit Complexity Lecture notes - [Uri Zwick] - http://cs.tau.ac.il/~zwick
53.15.2 Lecture notes - [Nader Bshouty] - http://cs.technion.ac.il/~bshouty

---

53.16 (THEORY) P/poly, Percolation and PARITY as 3-CNFSAT in 53.15

53.16.1 NP in P/poly (NP having polynomial size circuits) implies AM=MA and PH collapses to sigma2 /\ pi2.
53.16.2 PARITYSAT in 53.15 which is a special case of NP-complete has superpolynomial size circuits.
53.16.3 If Percolation is in NC/poly and both LHS and RHS of P(Good) converge to 100% noise stability, does it imply

```
arbitrary circuit size in RHS? In other words what does it imply when 2 circuits of varying size have equal stability

-----------------------------------------------------------------------------------------------------
53.16.3.1 If equal error (NoiseStability +/- delta) with different circuit sizes implies a lowerbound:
-----------------------------------------------------------------------------------------------------
There is an inherent conflict between 53.16.1, 53.16.3 which are conditions for polynomial size circuits for NP and 5
        - LHS of P(Good) is percolation NC/poly circuit in 100% noise stability regime.
        - RHS of P(Good) is non-percolation circuit of arbitrary size (e.g. DC circuit of exponential size , superpol
        - LHS NC/poly 100% noise stability implies a lowerbound when RHS (e.g superpolynomial size NP circuit from 53
        - the individual voter error/stability cannot be uniform 0.5 or 1 across the board for all voters - voters ha
        - varied stability/sensitivity per voter implies RHS majority voting is Poisson Distribution.
        - for probabilities other than 0.5 and 1, RHS P(good) summation becomes hypergeometric series requiring non-t
        - This assumes that RHS does not have a boolean function that is 100% noise stability regime similar to LHS P
        - Superpolynomial size for NP-complete PARITY3SAT (kind of a promise problem) implies P != NP and P != NP imp
In other words a democratic decision making which does not involve percolation is never perfect and thus BKS conjectu


-----------------------------------------------------------------------------------------------------
53.16.3.2 If equal error (NoiseStability +/- delta) with different circuit sizes doesn't imply a lowerbound:
-----------------------------------------------------------------------------------------------------
        - LHS of P(Good) is percolation NC/poly circuit in 100% noise stability regime.
        - RHS of P(Good) is of arbitrary size (e.g. superpolynomial size NP circuit from 53.16.2).
        - LHS NC/poly 100% noise stability doesn't imply a lowerbound when RHS (e.g superpolynomial size NP circuit f

-----------------------------------------------------------------------------------------------------
53.17 (THEORY) Oracle results and P=NP and PH
-----------------------------------------------------------------------------------------------------
From [Baker-Gill-Solovay] there are oracles A and B such that P^A = P^B and P^A != P^B implying natural proofs involv

-----------------------------------------------------------------------------------------------------
53.18 (THEORY) P(Good) circuit special case counterexample in PH collapse
-----------------------------------------------------------------------------------------------------
LHS of P(Good) - a sigma(p,k) hard (could be dictatorial or PRG) circuit with 100% (noise stability +/- delta) (assum
RHS of P(Good) - a sigma(p,k+1) hard majority voting circuit with 100% (noise stability +/- delta) (assuming such exi

If above converging P(good) LHS and RHS imply a lowerbound , LHS sigma(p,k) lowerbounds RHS sigma(p,k+1) implying PH
The matrix of noise versus BPP in 53.14 and 14 imply noise and BPP intersect, but if only classes in (sigma(p,k) - BP

Equating sigma(p,k) with sigma(p,k+l) is probably the most generic setting for P(Good) circuits discounting arithmeti

Example sigma(p,k)-complete QBFSAT(k) boolean function:
        there exists x1 for all x2 there exists x3 ... QBF1(x1,x2,x3,...,xk)

Example sigma(p,k+1)-complete QBFSAT(k+1) boolean function:
        there exists x1 for all x2 there exists x3 ... QBF2(x1,x2,x3,...,xk,x(k+1))

-----------------------------------------------------------------------------------------------------
53.19 (THEORY) Depth Hierarchy Theorem for P(Good) circuits
-----------------------------------------------------------------------------------------------------
Average case Depth Hierarchy Theorem for circuits by [Rossman-Servedio-Tan] - http://arxiv.org/abs/1504.03398 - state

-----------------------------------------------------------------------------------------------------
53.20 (THEORY) Ideal Voting Rule
-----------------------------------------------------------------------------------------------------
[Rousseau] theorem states that an ideal voting rule is the one which maximizes number of votes agreeing with outcome

-----------------------------------------------------------------------------------------------------
53.21 (THEORY) Plancherel's Theorem, Stability polynomial and Lowerbounds in 53.18
-----------------------------------------------------------------------------------------------------
Stability polynomial can be applied to QBFSAT(k) and QBFSAT(k+1) in 53.18.

By Plancherel's Theorem, Stability of a boolean function with noise rho is written as polynomial in rho:
        Stability(f) = sigma(rho^|S|*f(S)^2) , S in [n]
which can be equated to 1 to find roots of this polynomial - noise probabilities - where 100% stability occurs.

From HLMN theorem for sigma(p,k)-complete QBFSAT(k):
        sigma(f(A1)^2) <= 2*(M1)*2^(-t1^(1/d1)/20)  , |A1| > t1

From HLMN theorem for sigma(p,k+1)-complete QBFSAT(k+1):
        sigma(f(A2)^2) <= 2*(M2)*2^(-t2^(1/d2)/20) , |A2| > t2

When noise stability is 100%:
        Stability(QBFSAT(k)) = sigma(rho1^|S1|*f(S1)^2) = 1, S1 in [n]
        Stability(QBFSAT(k+1)) = sigma(rho2^|S2|*f(S2)^2) = 1, S2 in [n]
where rho1 and rho2 are variables each in the polynomial for QBFSAT(k) and QBFSAT(k+1). Solving these polynomials giv

Above deduction on PH collapse and possible PH-completeness contradicts 53.15 counterexample for NP having superpolyn
        - Equal decision error ( stability +/- delta ) does not imply lowerbound.
        - Roots of sigma(rho^|S|*f(S)^2) - 1 = 0 can only be complex with Re+Im parts and can't be real.
        - sigma(rho^|S|*f(S)^2) - 1 = 0 is zero polynomial where all coefficients are zero.
        - PH-collapse does not imply PH-completeness.
```

Stability polynomial above with rho as variable can be applied to any circuit to find the noise probabilities where 1

54(THEORY). Would the following experimental gadget work? It might, but impractical:
A voter gadget spawns itself into liker, disliker and neutral parallel threads on an entity to be judged. The neutral

55(THEORY). Evocation is a realworld everyday phenomenon - a word reminding of the other. Positive thought chooses th

56(THEORY). A crude way to automatically disambiguate is to lookup the corresponding class stack and analyze only bou

57(THEORY). Thus Sentiment Mining techniques can be used for the above hypergraph to get the overall sentiment of the

58(THEORY). Thus not only documents but also events in human life can be modelled as above hypergraph to get a graphi

##
59. Initial Design Notes for - Mundane Predictive Model:
##

- (FEATURE - DONE) DecisionTree, NaiveBayes and SVM classifiers and horoscope encoders are already in the AstroInfer
- (FEATURE - DONE) Encode the dataset which might be USGS or **NO**AA(Science on a Sphere) datasets or anyother dataset a
- (FEATURE - DONE) Above gives encoded horoscope strings for all classes of mundane events in both Zodiacal and Ascen
- (FEATURE - DONE) Above set is a mix of encoded strings for all events which can be classified using one of the clas
- (FEATURE - DONE) For each class the set of encoded horo strings in that class can be pairwise or multiple-sequence
- (FEATURE - DONE) There are two sets for each class after running the classifiers - one set is AscendantRelative and
- (FEATURE - DONE) The above steps give the mined astronomical patterns for all observed mundane events - Pattern_Mun

60. (FEATURE - DONE) Above has implemented a basic Class and Inference Model that was envisaged in 2003. Class is the

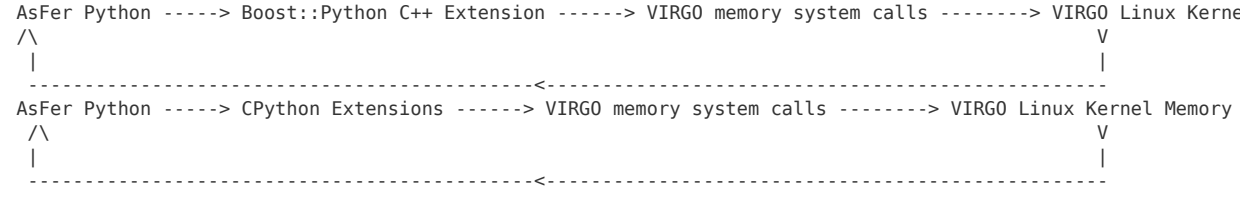
61. (THEORY) Now on to mining the classics for patterns: For all classes of events, running the classifier partitions

62. (THEORY) Thus correlation of the two sets *_Observed_* and *_Classic_* (each set has 2 subsets) for percentage si

63. (THEORY and IMPLEMENTATION) More importantly, for example, if a non-statistical,astrological model of rainfall i

64. (FEATURE - DONE-related to 65 and 139) Integrate AstroInfer,USB-md,KingCobra and VIRGO into an approximately inte
USB-md Design Document: http://*sourceforge.net/p/usb-md/code-0/HEAD/tree/USBmd_notes.txt*
VIRGO Design Document: http://*sourceforge.net/p/virgo-linux/code-0/HEAD/tree/trunk/virgo-docs/VirgoDesign.txt*
KingCobra Design Document: http://*sourceforge.net/p/kcobra/code-svn/HEAD/tree/KingCobraDesignNotes.txt*
This AsFer+USBmd+VIRGO+KingCobra would henceforth be known as "Krishna iResearch Intelligent Cloud OS - NeuronRain "

65. (THEORY and IMPLEMENTATION) Related to point 64 - Software Analytics - source code and logs analytics - related t



References:

65.1 Analytics of Device Driver code - http://*research.microsoft.com/en-us/groups/sa/drivermine_asplos14.pdf*
65.2 Logging - http://*research.microsoft.com/en-US/groups/sa/loggingpractice.pdf*
65.3 Law of leaky abstraction - http://*www.joelonsoftware.com/Articles/LeakyAbstractions.html*
65.4 Function Point Analysis - http://*www.bfpug.com.br/Artigos/Albrecht/MeasuringApplicationDevelopmentProductivity.*
65.5 Function Point Analysis and Halstead Complexity Measures - https://*en.wikipedia.org/wiki/Halstead_complexity_me*
65.6 Cyclomatic Complexity of a Program - Euler Characteristic of program flow graph (E - V + 2) - Approximate code

66. (THEORY) Encoding a document with above Hypergraph of Class stack vertices and Hyperedges - This hypergraph of cc

67. An automaton or Turing machine model for data or voice (or musical) samples - Voice samples or musical notations

(FEATURE - DONE) 68. Music Pattern Mining - Computational Music - Alternatively, a Discrete Fourier Transform on a se
----
References:

68.1 https://*ccrma.stanford.edu/~jos/st/*

69. (FEATURE - Audacity FFT - DONE) Music Pattern Mining - Experimental Idea of previous two points is to mine patter
        69.1 FFT of the two audio files were done by Audacity and frequency domain files were obtained with sampling
        69.2 FFTs of these two have been committed in music_pattern_mining/. Similarity is observed by peak decibel f

---

```
70-79 (THEORY - EventNet Implementation DONE) EventNet and Theoretical analysis of Logical Time:
-----------------------------------------------------------------------------------------------

70. In addition to Concept wide hypergraph above, an interesting research could be to create an EventNet or events cc

71. Each event node in the node which is a set as above, can have multiple outcomes and hence cause multiple effects.

72. If the EventNet is formulated as a Dynamic Graph with cycles instead of Static Graph as above where there causal

73. EventNet can be partitioned into "Past","Present" and "Future" probably by using some MaxFlow-MinCut Algorithms.

74. A recreational riddle on the EventNet: Future(Past) = Present. If the Future is modelled as a mathematical functi

75. Conjecture: Undirected version of EventNet is Strongly Connected or there is no event in EventNet without a cause

76. Events with maximum indegree and outdegree are crucial events that have deep impact in history.

77. Events in each individual entity's existence are subgraphs of universal EventNet. These subgraphs overlap with ea

78. There is an implicit time ordering in EventNet due to each causation edge. This is a "Logical Clock" similar to L

79. EventNet can also be viewed as a Bayesian Network.

--------------------------------
80. (THEORY) Mining EventNet for Patterns:
--------------------------------
As an old saying goes "history repeats itself". Or does it really? Are there cycles of events? Such questions warrant

Frequent Subgraph Mining of above EventNet (i.e whether a subgraph "repeats" with in the supergraph) as mentioned in

There is a striking resemblance of EventNet to sequence mining algorithm CAMLS - a refined Apriori algorithm - (http:

----------------------------
81-86. (THEORY) Fractal nature of events:
----------------------------
81. Are events self-similar or exhibit fractal nature? This as in item 71, needs mining the event nodes which are set

82. EventNet is a Partial Order where there may not be edges of causality between some elements. Thus a Topological S

83. EventNet on history of events in the universe is quite similar to Directed Acyclic Graph based Scheduling of task

84. In the above EventNet, each node which is a set can indeed be a graph also with freewill interactions amongst the

85. Similar to Implication graphs as Random Growth Graphs, EventNet also qualifies as a Random Growth Network as even

86. Regularity Lemma can be used as a tool to test Implication and EventNet Random Growth Graphs - specifically seems

*****************************************************************************************************
87.                                    COMMIT RELATED NOTES
*****************************************************************************************************

(FEATURE- DONE) commits as on 3 September 2013
------------------------------
DecisionTree, NaiveBayes and SVM classifier code has been integrated into AstroInfer toplevel invocation with boolean

(FEATURE- DONE) commits as on 6 September 2013
------------------------------
In addition to USGS data, NOAA Hurricane dataset HURDAT2 has been downloaded and added to repository. Asfer Classifie

(FEATURE- DONE) commits as on 12 September 2013
------------------------------
1.asfer_dataset_segregator.py and asfer_dataset_segregator.sh - Python and wrapper shell script that partitions the p

2.DateTimeLongLat data for all datasets within an event class (text file) need to be collated into a single asfer.enc

(FEATURE- DONE) commits as on 2 November 2013
------------------------------
Lot of commits for implementation of Discrete Hyperbolic Factorization with Stirling Formula Upperbound (http://sourc
        1) Multiple versions of Discrete Hyperbolic Factorization uploaded in http://sites.google.com/site/kuja27/
        2) Handwritten by myself - http://sourceforge.net/projects/acadpdrafts/files/DiscreteHyperbolicFactorization
        3) Latex version - http://sourceforge.net/projects/acadpdrafts/files/DiscreteHyperbolicPolylogarithmicSieveF
)

(FEATURE- DONE) commits as on 20 November 2013 and 21 November 2013
--------------------------------------------------
1. Lots of testing done on Discrete Hyperbolic Factorization sequential implementation and logs have been added to re

2. An updated draft of PRAM NC version of Discrete Hyperbolic Factorization has been uploaded at:
http://sourceforge.net/projects/acadpdrafts/files/DiscreteHyperbolicPolylogarithmicSieveForIntegerFactorization_PRAM_

3. Preliminary Design notes for CRCW PRAM implementation of the above is added to repository at: http://sourceforge.n
```

```
1759
1760
1761  ------------------------------
1762  88.Tagging as on 16 December 2013
1763  ------------------------------
1764  AsFer version 12.0 and VIRGO version 12.0 have been tagged on 6 December 2013 (updated the action items above).
1765
1766  (THEORY) 89. More reference URLs for Parallel RAM design of above NC algorithm for Discrete Hyperbolic Factorization
1767  ------------------------------------------------------------------------------------------------------------------------
1768  1.http://arxiv.org/pdf/1202.6575.pdf?origin=publication_detail
1769  2.https://electures.informatik.uni-freiburg.de/portal/download/3/6951/thm15%20-%20parallel%20merging.ppt (Ranks of el
1770  3.http://cs.brown.edu/courses/csci2560/syllabus.html (Lecture Notes on CREW PRAM and Circuit Equivalence used in the
1771  Hyperbolic Factorization above - http://cs.brown.edu/courses/csci2560/lectures/lect.24.pdf)
1772  4.http://cs.brown.edu/courses/csci2560/lectures/lect.22.ParallelComputationIV.pdf
1773  5.http://www.cs.toronto.edu/~bor/Papers/routing-merging-sorting.pdf
1774  6.http://www.compgeom.com/~piyush/teach/AA09/slides/lecture16.ppt
1775  7.Shift-and-subtract algorithm for approximate square root computation implemented in Linux kernel (http://lxr.free-e
1776  8.http://research.sun.com/pls/apex/f?p=labs:bio:0:120 - Guy Steele - approximate square root algorithm
1777  9.http://cstheory.stackexchange.com/questions/1558/parallel-algorithms-for-directed-st-connectivity - related theoret
1778  10.PRAM and NC algorithms - http://www.cs.cmu.edu/afs/cs/academic/class/15750-s11/www/handouts/par-notes.pdf
1779  11. An Efficient Parallel Algorithm for Merging in the Postal Model - http://etrij.etri.re.kr/etrij/journal/getPublis
1780  12. Parallel Merge Sort - http://www.inf.fu-berlin.de/lehre/SS10/SP-Par/download/parmerge1.pdf
1781  13. NC and PRAM equivalence - www.cs.tau.ac.il/~safra/ACT/CCandSpaceB.ppt
1782  14. Extended version of BerkmanSchieberVishkin ANSV algorithm - http://www1.cs.columbia.edu/~dany/papers/highly.ps.Z
1783  15. Structural PRAM algorithms (with references to NC) - http://www.umiacs.umd.edu/users/vishkin/PUBLICATIONS/icalp91
1784  16. Parallel Algorithms FAQ - NC, RAM and PRAM - Q24 - http://nptel.ac.in/courses/106102114/downloads/faq.pdf
1785
1786  17. Defintions related to PRAM model - http://pages.cs.wisc.edu/~tvrdik/2/html/Section2.html - Input to PRAM is N ite
1787
1788  18. Reduction from PRAM to NC circuits - http://web.cse.ohio-state.edu/~gurari/theory-bk/theory-bk-sevense6.html#Q1-8
1789
1790  19. Length of input instance in PRAM (n or N) - http://web.cse.ohio-state.edu/~gurari/theory-bk/theory-bk-sevense2.ht
1791
1792  20. Quoted abstract - "... The all nearest smaller values problem is defined as follows. Let A = (a1 ; a2 ; : : : ; a
1793
1794  21. SIMULATE_RAM subcircuit in point 18 - Number of bits allowed per PRAM cell in PRAM to NC reduction - http://hall.
1795
1796
1797  ------------------------------------------------------------------------------------------------------------------------
1798  (THEORY) 90.Some notes on extensions to Integer partitions and Hash Functions
1799  (https://sites.google.com/site/kuja27/IntegerPartitionAndHashFunctions.pdf?attredirects=0)
1800  ------------------------------------------------------------------------------------------------------------------------
1801  1. Riemann sums (discrete approximation of Riemann integral) of all the functions corresponding to the hash functions
1802
1803  2. Hardy-Ramanujan asymptotic bound for partition function p(n) is ~ O(e^(pi*sqrt(0.66*n))/(4*1.732*n) which places a
1804
1805  3. If m-sized subsets of the above O(m!*e^(sqrt(n))/n) number of hash functions are considered as a (k,u)-universal c
1806
1807  4. Thus deriving a bound for number of possible hash functions in terms of number of keys and values could have beari
1808
1809  5. Birthday problem and Balls and Bins problem - Since randomly populating m bins with n balls and probability of pec
1810
1811  6. Restricted partitions which is the special case of integer partitions has some problems which are NP-complete. Mor
1812
1813  7. The special of case of majority voting which involves integer partitions described in https://sites.google.com/sit
1814
1815  8. Thus Majority voting can be shown to be NP-complete in 2 ways:
1816         8.1 by Democracy circuit (Majority with SAT) in http://sourceforge.net/projects/acadpdrafts/files/Implicatior
1817         8.2 by reduction from an NP-hard instance of Restricted Partition problem like Money changing problem for Maj
1818
1819  9. Infact the above two ways occur in two places in the process of democratic voting: The democracy circuit is needed
1820
1821  10. Point 8.2 above requires a restricted partition with distinct non-repeating parts. There are many results on this
1822
1823  11. An interesting manifestation of point 10 is that nothing in real-life voting precludes a tie and enforces a restr
1824
1825  12. Counting Number of such restricted partitions is a #P-complete problem - https://www.math.ucdavis.edu/~deloera/TA
1826
1827  13. If a Hash table is recursive i.e the chains themselves are hashtables and so on... then this bijectively correspc
1828
1829  14. If the hash table chains are alternatively viewed as Compositions of an integer (ordered partitions) then there a
1830
1831  15. In the summation over all parts of partitions derived in https://sites.google.com/site/kuja27/IntegerPartitionAnd
1832
1833  16. Logarithm of above summation then is equal to (n-1) and thus can be equated to any partition of n. Thus any parti
1834
1835  ------------------------------------------------------------------------------------------------------------------------
1836  91. Updated drafts on Integer partitions and hash function (with points in 80 above) , Circuits for Error probability
1837  ------------------------------------------------------------------------------------------------------------------------
1838  1. https://sites.google.com/site/kuja27/IntegerPartitionAndHashFunctions_2014.pdf?attredirects=0&d=1
1839  2. https://sites.google.com/site/kuja27/CircuitForComputingErrorProbabilityOfMajorityVoting_2014.pdf?attredirects=0&d
```

```
1840
1841  ------------------------------------------------------------------------------
1842  92. Reference URLs for restricted integer partitions with distinct parts
1843  ------------------------------------------------------------------------------
1844  (for http://sites.google.com/site/kuja27/IntegerPartitionAndHashFunctions_2014.pdf?attredirects=0&d=1)
1845  1. Generating function - http://math.berkeley.edu/~mhaiman/math172-spring10/partitions.pdf
1846  2. Schur's theorem for asymptotic bound for number of denumerants - http://en.wikipedia.org/wiki/Schur's_theorem
1847  3. Frobenius problem - http://www.math.univ-montp2.fr/~ramirez/Tenerif3.pdf
1848  4. Locality Sensitive Hashing (that groups similar keys into a collision bucket chain using standard metrics like Ham
1849  5. Locality Sensitive Hashing and Lattice sets (of the diophantine form a1*x1+a2*x2+...+an*xn) -  http://hal.inria.fr
1850  6. Minhash and Jaccard similarity coefficient - http://en.wikipedia.org/wiki/MinHash#Jaccard_similarity_and_minimum_h
1851
1852  ------------------------------------------------------------------------------
1853  (THEORY) 93. Reduction from Money Changing Problem or 0-1 Integer LP to Restricted Partitions with distinct parts
1854  ------------------------------------------------------------------------------
1855  If the denominations are fixed as 1,2,3,4,5,....,n then the denumerants to be found are from the diaphantine equation
1856  a1*1 + a2*2 + a3*3 + a4*4 + a5*5 + ...+ an*n
1857  (with ai = 0 or 1). GCD of all ai(s) is 1. Thus Schur's theorem for MCP or Coin Problem applies.
1858  Integet 0-1 LP NP-complete problem can also be reduced to above diophantine format instead of MCP. Finding one such d
1859  boolean values is then NP-complete and hence finding one partition with distinct non-repeating parts is NP-complete (
1860  majority vote).
1861
1862  ------------------------------------------------
1863  (FEATURE- DONE) 94. Commits as on 23 April 2014
1864  ------------------------------------------------
1865  Updated pgood.cpp with some optimizations for factorial computations and batched summation to circumvent overflow to
1866  For Big decimals IEEE 754 with 112+ bits precision is needed for which boost::multiprecision or java.math.BigDecimal
1867
1868  ------------------------------------------------
1869  (FEATURE- DONE) 95. Commits as on 7 July 2014
1870  ------------------------------------------------
1871  Initial implementation of a Chaos attractor sequence implementation committed to repository.
1872
1873  ------------------------------------------------
1874  (FEATURE- DONE) 96. Commits as on 8 July 2014
1875  ------------------------------------------------
1876  Python-R (rpy2) code for correlation coefficient computation added to above Chaos attractor implementation.
1877
1878  ------------------------------------------------------------------
1879  (FEATURE- DONE) 97. Time Series Analysis - Commits as on 9 July 2014
1880  ------------------------------------------------------------------
1881  DJIA dataset, parser for it and updated ChaosAttractor.py for chaotic and linear correlation coefficient computation
1882  have been committed.
1883
1884  ------------------------------------------------
1885  (FEATURE- DONE) 98. Commits as on 10 July 2014
1886  ------------------------------------------------
1887  Python(rpy2) script for computing Discrete Fourier Transform for DJIA dataset has been added (internally uses R). [py
1888
1889  ------------------------------------------------
1890  (FEATURE- DONE) 99. Commits as on 11 July 2014
1891  ------------------------------------------------
1892  Python(rpy2) script for spline interpolation of DJIA dataset and plotting a graph of that using R graphics has been a
1893
1894  ------------------------------------------------
1895  (FEATURE- DONE) 100. Commits as on 15 July 2014 and 16 July 2014
1896  ------------------------------------------------
1897  Doxygen documentation for AsFer, USBmd, VIRGO, KingCobra and Acadpdrafts have been committed to GitHub at https://git
1898
1899  ------------------------------------------------
1900  (FEATURE- DONE) 101.Commits as on 23 July 2014
1901  ------------------------------------------------
1902  Draft additions to http://arxiv.org/pdf/1106.4102v1.pdf are in the Complement Function item above. An R graphics rpy2
1903
1904  ------------------------------------------------------------------
1905  (FEATURE - THEORY - Visualizer Implementation DONE) 102. Dense Subgraphs of the WordNet subgraphs from Recursive Glos
1906  ------------------------------------------------------------------
1907  http://arxiv.org/abs/1006.4458 and http://www.nist.gov/tac/publications/2010/participant.papers/CMI_IIT.proceedings.p
1908          102.1 http://kylescholz.com/projects/wordnet/
1909          102.2 http://www.visuwords.com
1910          102.3 http://www.visualthesaurus.com/browse/en/Princeton%20WordNet
1911
1912  ------------------------------------------------------------------
1913  103. AsFer version 14.9.9 release tagged on 9 September 2014
1914  ------------------------------------------------------------------
1915  ------------------------------------------------------------------
1916  (FEATURE- DONE) 104. Commits as on 9 October 2014
1917  ------------------------------------------------------------------
1918  Initial python script implementation for Text compression committed with logs and screenshot. Decompression is still
1919
1920  ------------------------------------------------------------------
```

```
(FEATURE- DONE) 105. Commits as on 21 October 2014
----------------------------------------------------------------------
An experimental POC python implementation that uses Hidden Markov Model for decompression has been committed. But it

----------------------------------------------------------------------
(THEORY) 106. Creating a summary graph from EventNet
----------------------------------------------------------------------
EventNet graph for historical cause and effect described above can be huge of the order of trillion vertices and edge
events might be necessary at times - similar to a document summarization. Summarizing a graph of event cause-effects

----------------------------------------------------------------------
(FEATURE- DONE) 107. Commits as on 1 November 2014
----------------------------------------------------------------------
A minimal PAC learning implementation in python to learn a Boolean Conjunction from dataset has been added to reposit

----------------------------------------------------------------------
(FEATURE- DONE) 108. Commits as on 4 November 2014
----------------------------------------------------------------------
Initial implementation for Minimum Description Length has been added to repository.

----------------------------------------------------------------------
(FEATURE- DONE) 109. Commits as on 6 November 2014
----------------------------------------------------------------------
Python Shannon Entropy implementation for texts has been added to repository and is invoked in MinimumDescLength.py M

----------------------------------------------------------------------
(FEATURE- DONE) 110. Commits as on 7 November 2014
----------------------------------------------------------------------
Python Kraft Inequality MDL implementation has been committed.

----------------------------------------------------------------------
(FEATURE- DONE) 111. Commits as on 11 November 2014
----------------------------------------------------------------------
C++ implementation for Wagner-Fischer Dynamic Programming algorithm that iteratively computes Levenshtein Edit Distan

----------------------------------------------------------------------
(FEATURE - DONE) 112. Expirable objects
----------------------------------------------------------------------
Setting expiry to an object is sometimes essential to control updates and access to an object. Example application of
should be accessible or displayable for only a finite number of times and after that it has to self-destruct. Though
straightforward with operator overloading, in C it looks non-trivial. Presently only C++ implementation is added to r

------------------------------------------------------------------------------
(FEATURE- DONE) 113. Commits as on 11 November 2014 - Expirable template class implementation
------------------------------------------------------------------------------
Similar to weak_ptr and shared_ptr, a generic datatype that wraps any datatype and sets an expiry count to it has bee
in cpp-src/expirable/ . For example, a bitmap or a JPEG image can be wrapped in expirable container and it can be acc
If expiry count is 1, the object (an image for example) can be written to or displayed only once and thus a singleton
only for rvalues. Implementing for lvalues (just a usage without assignment should be able to expire an object) seems
there is no operator overloading available for lvalues - i.e there is no "access" operator to overload. This might be
MAC electronic money also.(std::move() copy-by-move instead of a std::swap() copy-by-swap idiom)

------------------------------------------------------------------------------
(FEATURE- DONE) 114. Commits as on 12 November 2014
------------------------------------------------------------------------------
Expirable template in asferexpirable.h has been updated with 2 operator=()
functions for copy-assignment and move-assignment which interally invoke
std::swap() and std::move(). Explicit delete(s) are thus removed. Also example
testcase has been updated to use a non-primitive datatype.

------------------------------------------------------------------------------
(FEATURE- DONE) 115. Commits as on 13 November 2014
------------------------------------------------------------------------------
Overloaded operator*() function added for expiry of "access" to objects. With this problem mentioned in 113 in tracki
circumvented indirectly.

------------------------------------------------------------------------------
(FEATURE- DONE) 116. Commits as on 15 November 2014
------------------------------------------------------------------------------
Python implementation for Perceptron and Gradient has been added to repository.

------------------------------------------------------------------------------
(FEATURE- DONE) 117. Commits as on 17 November 2014
------------------------------------------------------------------------------
Python implementation for Linear and Logistic Regression in 2 variables.

------------------------------------------------------------------------------
(FEATURE- DONE) 118. Commits as on 20 November 2014
------------------------------------------------------------------------------
C++ implementation of K-Means clustering with edit distance as metric has been added to
repository.
```

```
2002
2003  ------------------------------------------------------------------------
2004  (FEATURE- DONE) 119. Commits as on 21 November 2014
2005  ------------------------------------------------------------------------
2006  C++ implementation of k-Nearest Neighbour clustering has been added to repository.
2007
2008  ------------------------------------------------------------------------
2009  120. Commits as on 24 November 2014
2010  ------------------------------------------------------------------------
2011  Bugfixes to kNN implementation.
2012
2013  ------------------------------------------------------------------------
2014  (FEATURE- DONE) 121. Commits as on 25 November 2014
2015  ------------------------------------------------------------------------
2016  Python script for decoding encoded horoscope strings (from Maitreya's Dreams) has been added to
2017  repository.
2018
2019  ------------------------------------------------------------------------
2020  122. (FEATURE - DONE) Commits as on 3 December 2014
2021  ------------------------------------------------------------------------
2022  Initial implementation for EventNet:
2023  ------------------------------------
2024  1. EventNet python script has inputs from two text files - EventNetEdges.txt and EventNetVertices.txt - which define
2025  2. This uses GraphViz (that in turn writes a dot file) and python-graph+gv packages
2026  3. GraphViz writes to EventNet.graphviz.pdf and EventNet.gv.png rendered and visualized graph files.
2027  4. Above text files are disk-stored and can be grown infinitely.
2028  5. EventNetVertices.txt has the format:
2029         <event vertex> - <csv of partakers in the event vertex>
2030     EventNetEdges.txt has the format:
2031         <ordered pairs of vertices for each causation edge>
2032  6. Topological Sorting of EventNet using python-graph algorithms package
2033
2034  ----------------------------------------------------------------------------------------
2035  (THEORY) 123. EventNet - partakers of events and an application of EventNet to a related problem
2036  ----------------------------------------------------------------------------------------
2037  Event vertices have partakers of event as defined in 122. Point 84-86 previously defined EventNet as
2038  a fractal graph tensor where each vertex is a graph or partakers. Alternative formulation of this is
2039  where the partakers are just key words or persons in that event. For example, a fictitious story can be translated in
2040
2041  ----------------------------------------------------------------------------------------
2042  (FEATURE- DONE) 124. Commits as on 4 December 2014
2043  ----------------------------------------------------------------------------------------
2044  EventNet - a cloudwide event ordering with unique id implementation
2045  -------------------------------------------------------------------
2046  EventNet has 2 input files for vertices and edges with partakers and
2047  writes an output file with ordering of the events
2048
2049  Input - EventNetVertices.txt has the format:
2050         <event vertex> - <csv of partakers> - <tuples of conversations amongst the partakers # separated>
2051         partakers could be machine id(s) or IP addresses and thread id(s) and the conversations being the        messa
2052         within each event vertex
2053
2054  Input - EventNetEdges.txt has the format:
2055         <event vertex1, event vertex2>
2056
2057  Output - EventNetOrdering.txt has the format:
2058         <index in chronologically ascending order> - <event id>
2059
2060  EventNet script thus is run in a central node which has the input files above that is
2061  updated by all the nodes in cloud. Outgoing edge from an event vertex has partakers from multiple
2062  events and thus is an outcome of the event. If the input files are split and stored in multiple cloud
2063  nodes, the topological sorts for multiple input files have to be merged to create a single cloudwide
2064  ordering.
2065
2066  ----------------------------------------------------------------------------------------------------
2067  (THEORY) 125. Massive EventNet computation
2068  ----------------------------------------------------------------------------------------------------
2069  Each conversation of the events needs to create a log message that is sent to the EventNet service
2070  which updates the input vertices and edges files. The python EventNet script run optionally on a hadoop
2071  cluster mapreduce recomputes the topological ordering periodically. This is quite tedious a process that can flood th
2072
2073  ----------------------------------------------------------------------------------------------------
2074  (FEATURE- DONE) 126. Commits as on 5 December 2014
2075  ----------------------------------------------------------------------------------------------------
2076  Initial C++ Boost::graph based implementation for EventNet has been added to repository.
2077
2078  ----------------------------------------------------------------------------------------------------
2079  (THEORY) 127. 2-dimensional random walks for decision making (experimental)
2080  ----------------------------------------------------------------------------------------------------
2081  Continued from point 49 above, the psychological process of decision making (in a mental conflict with
2082  two opposing directions) is akin to a game theoretical notion of Nash Equilibrium - where a huge payoff
```

matrix is constructed for the random walk directions as strategies for the two conflicting decisions. There could be

---

128. Commits as on 9 December 2014

Bugfixes for DOT file generation and toposort in C++ EventNet implementation.

---

129. (THEORY) Draft Updates to https://sites.google.com/site/kuja27/CircuitForComputingErrorProbabilityOfMajorityVoti

---

For even number of finite population (electorate) the binomial coefficient summation in uniform distribution has an e

        (2^n - nC(n/2))/2^(n+1) = 0.5 - epsilon where epsilon=sqrt(1/(2*n*pi)) limit for which vanishes at infinity

(or) probability of good choice is epsilon less than 50% for finite even number of electorate for uniform distributic

If this infinite-ness breaches the polynomiality then what is quite puzzling is that RHS becomes a Direct Connect DC

"...
6.6
Circuits of exponential size
As noted, every language has circuits of size O(n^2^n). However, actually finding these circuits may
be difficult��? sometimes even undecidable. If we place a uniformity condition on the circuits, that
is, require them to be efficiently computable then the circuit complexity of some languages could
exceed n^2^n . In fact it is possible to give alternative definitions of some familiar complexity classes,
analogous to the definition of P in Theorem 6.7.
Definition 6.28 (DC-Uniform)
Let {C n } n���1 be a circuit family. We say that it is a Direct Connect uniform (DC uniform) family if,
given hn, ii, we can compute in polynomial time the i th bit of (the representation of) the circuit C n .
More concretely, we use the adjacency matrix representation and hence a family {C n } n���N is DC
uniform iff the functions SIZE, TYPE and EDGE defined in Remark ?? are computable in polynomial
time.
Note that the circuits may have exponential size, but they have a succinct representation in
terms of a TM which can systematically generate any required node of the circuit in polynomial
time.
Now we give a (yet another) characterization of the class PH, this time as languages computable
by uniform circuit families of bounded depth. We leave it as Exercise 13.

Theorem 6.29
L ��? P H iff L can be computed by a DC uniform circuit family {C n } that
��? uses AND, OR, **NO**T gates.
O(1)
��? has size 2^n
and constant depth (i.e., depth O(1)).
��? gates can have unbounded (exponential) fanin.
��? the **NO**T gates appear only at the input level.
If we drop the restriction that the circuits have constant depth, then we obtain exactly EXP
..."

The RHS Majority+SAT circuit has all characteristics satisfying the DC-uniformity in the theorem above - size can be

It is not necessary that per voter SAT is same for all voters. Each voter can have unrestricted depth SAT clauses (in

even if a BQP algorithm is used in voting outside the purview of PH but in EXP, it doesn't change the above unless:
 - perfection is impossible i.e there cannot be zero-error processes in the universe
 - DC-circuit is not drawable (or undecidable if it can be constructed)
 - infinite majority is undecidable (so circuit is DC non-uniform and not DC-uniform)
 - the voter CNF can only be 2-SAT (which is highly unlikely) and not k-SAT or 3-SAT

129.1 Toda's theorem and P(good) circuit above:

PH is contained in P^*#P (or) P with #no-of-solutions-to-SAT oracle (Toda's theorem).*
If zero-error Majority+SAT voting DC uniform circuit is in PH then due to LHS=RHS of the P(good) series convergence (
PH collapses(?) to P (quite unbelievably):
LHS Pseudorandom choice is in P while RHS Majority+SAT is in PH=DC circuit (restricted depth) or EXP (unrestricted de
(i.e there is a P algorithm for PH).
if P=PH:
  P=PH is in P^*#P by Toda's theorem*

if P=EXP:
 P=PH=EXP in P^*#P or P=PH=EXP=P^#P (which is a complete collapse)*

[p=0.5 is the uniform distribution which is a zero bias space while for other probabilities some bit patterns are les

---

130. (THEORY) Counterexample definition for P Vs NP

Majority Circuit with SAT voter inputs with finite odd number of voters, in uniform distribution
converges to 1/2 same as LHS for pseudorandom choice. Even number of voters is a special case described

previously. Also both LHS and RHS converge to 1 if probability is 1 (without any errors in pseudorandom choice and ma

---
131. (THEORY) Infinite majority and SAT+Majority Circuit

Infinite version of majority circuit is also a kind of heavy hitters problem for streaming algorithms where majority

---
132. (THEORY) May's Theorem of social choice

May's Theorem: In a two-candidate election with an odd number of voters, majority rule is the only voting system that

May's theorem is 2-candidate analog of Arrow's Theorem for 3-candidate condorcet voting. May's theorem for 2 candidat

Additional References:

132.1 May's theorem - http://*www.math.cornell.edu/~mec/Summer2008/anema/maystheorem.html*

---
133. (THEORY) Pseudorandom number generator and Majority voting for choice on a set
---
Let S be a set of elements with "goodness" attribute for each element. Choice using LHS and RHS on this set is by a P
133.1 Accuracy of the PRG - how random or k-wise independent the PRG is - this is usually by construction of a PRG (E
133.2 Goodness of chosen element - PRG is used to choose an element from the set - this mimicks the realworld phenome
133.3 Thus if the set S is 100% perfect - all elements in it are the best - LHS of P(Good) is 1. Similarly the RHS is

From the aforementioned,it is evident that for a 100% perfect set,both PRG(LHS) and Majority+SAT(RHS) Voting are two

---
(FEATURE- DONE) 134.Commits as on 8 January 2015

134.1 C++ implementation for Longest Common Substring has been added to repository with logs. Datasets were the clust
134.2 AsFer+USBmd+VIRGO+KingCobra+Acadpdrafts - version 15.1.8 release tagged.

---
135. (THEORY) Updates, Corrigenda and References to Space Filling Algorithm in :
https://*sites.google.com/site/kuja27/Analysis%20of%20a%20Randomized%20Space%20Filling%20Algorithm%20and%20its%20Linea*

135.1 The standard LP form can be obtained by slack variables (with only equalities) - mentioned as free variables in

135.2 Pseudorandom generator with linear stretch in NC1 - http://*homepages.inf.ed.ac.uk/mcryan/mfcs01.ps*
- ... " The notion of deterministically expanding a short seed into a long string that ... the question of whether st

135.3 This space filling algorithm is in NC (using Tygar-Rief) with an assumption that multiplicative inverse problem

135.4 Tygar-Rief algorithm outputs n^c bits in parallel (through its PRAM processors) for n=logN for some composite N

135.5 Above pseudorandom bits have to be translated into the coordinate positions in the grid which is Z. In other wo

135.6 Constant c can be suitably chosen as previously to be c = log(Z*logZ)/logn (n need not be equal to Z)

135.7 Either the n^c sized pseudorandom string can be split sequentially into (logZ/2+logZ/2) sized substring coordin

135.8 In both splits above, same coordinate tuple might get repeated - same bit position can be set to 1 more than on

135.9 There are Z substrings created out of the 2^n^c pseudorandom strings generated by Tygar-Rief Parallel PRG. Henc

135.10 Maximizing the LP can be reduced to minimizing the collisions (or) repetitive coordinate substrings above thus

135.11 The grid filling can be formulated as a Cellular Automaton of a different kind - by setting or incrementing th

135.12 Instead of requiring the non-repetitive coordinate substrings mentioned in (10) supra, the grid filling can be

---
135.13 Cellular Automaton Algorithm:

(Reference for reductions below: http://*cstheory.com/stockmeyer@sbcglobal.net/csv.pdf [Chandra-Stockmeyer-Vishkin])
[http://*sourceforge.net/p/asfer/code/HEAD/tree/asfer-docs/AstroInferDesign.txt and*
https://*github.com/shrinivaasanka/asfer-github-code/blob/master/asfer-docs/AstroInferDesign.txt]*

The circuits described in [ChandraStockmeyerVishkin] are constant depth circuits [AC0] and thus in [NC1] (unbounded

From wikipedia - Linear programs are problems that can be expressed in canonical form:
c^T * X
subject to AX <= b
and X >= 0

maximize X
subject to AX <= some maximum limit
and X >= 0

```
Below Cellular Automaton Monte Carlo NC algorithm for grid filling assumes that:
C=[1,1,1,...1]
A=[1,1,1....1] in the above and the vector of variables X is mapped to the grid - sequentially labelled in ascending


Grid cells (or variables in above grid) are initialized to 0.

loop_forever
{
        (135.13.1) Parallel Pseudorandom Generator (Tygar-Rief) outputs bit string which are split into coordinates c
        (135.13.2) Each grid cell has a constant depth Comparison Circuit that outputs 1 if the grid cell value excee
        (135.13.3) If above Comparison gate outputs 1, then another NC circuit can be constructed that finds the mini
}

Above can be diagramatically represented as: (for N=a*b cells in grid in parallel)
--------------------------------------------------------------------------------

                cell(1,1)       ..............     cell(a,b)
                ---------------------------------------------
                PRG in parallel         ....    PRG in parallel
                      ||                                ||
                Comparison              ....       Comparison
                      ||                                ||
        Minimum of 8 neighbours         ....      Minimum of 8 neighbours
                      ||                                ||
                Increment                           Increment
                and                                 and
                Decrement                           Decrement

135.14 For example, a 3*3 grid is set by pseudorandom coordinate substrings obtained from Parallel PRG (6,7,7,8,9,1,3

                1  1  2
                0  0  1
                2  1  1

Above can be likened to a scenario where a viscous fluid is unevenly spread at random from heavens.
By applying cellular automaton NC circuit algorithm above, grid becomes: (minimizes collisions,maximizes variables an

                1  1  1
                1  1  1
                1  1  1

Above can be likened to a scenario where each 8-neighbour grid cell underneath computes the evened-out cellular autom
Thus there are 2 NC + 1 AC circuits which together maximize the special case of P-Complete LP instance without simple

------------------------------------------------------------------------------------------------
135.15 References on Cellular Automata, Parallel Computation of CA, Fluid Dynamics, Navier-Stokes equation, Percolati
------------------------------------------------------------------------------------------------
135.15.1 http://www.nt.ntnu.no/users/skoge/prost/proceedings/acc11/data/papers/1503.pdf
135.15.2 http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.52.8377
135.15.3 http://www.stephenwolfram.com/publications/academic/cellular-automaton-fluids-theory.pdf
135.15.4 Constant Depth Circuits - http://www.cs.berkeley.edu/~vazirani/s99cs294/notes/lec5.ps
135.15.5 Complexity Theory Companion - https://books.google.co.in/books?id=ysu-bt4UPPIC&pg=PA281&lpg=PA281&dq=Chandra
135.15.6 Pseudorandom Permutation in Parallel - https://www.nada.kth.se/~johanh/onewaync0.ps

------------------------------------------------------------------------------------------
135.16 A related algorithm - filling a square with infinitely many inscribed circles
------------------------------------------------------------------------------------------
Let S be a square of unit area. This square is filled with infinitely many circles of varying radii. This
can be expressed in terms of geometric equations for circle - x(i)^2 +y(i)^2 = r(i)^2. Each (x(i),y(i)) is a coordina

135.17 Above grid can be alternatively formulated as a Voronoi Diagram where Parallel PRG randomly sets multiple poin

References:
----------
135.18. Circle Packing and Grid Filling - http://11011110.livejournal.com/332331.html - Filling a square with non-ove
135.19. 135.16 is a slight variant of Kepler's Theorem [Proof - [Hales] - https://sites.google.com/site/thalespitt/]
135.20 Maximizing Sum of Radii of Balls - [Eppstein] - http://www.ics.uci.edu/~eppstein/pubs/Epp-CCCG-16-slides.pdf
135.21 Thue's Theorem For packing of discs on 2D - http://www.math.stonybrook.edu/~tony/whatsnew/dec00/paper.html - (

--------------------------------------------------------------------------
(FEATURE- DONE) 136. Commits as on 28,29 January 2015
--------------------------------------------------------------------------
Python parser script to translate Longest Common Substring extracted from clustered or classified
set of encoded strings has been added to repository.

------------------------------------------------------------------------------------------
(FEATURE - DONE) 137. Mining the Astronomical Datasets using KMeans and kNN - sequence of algorithms in AstroInfer
- Commits as on 4 February 2015
------------------------------------------------------------------------------------------
```

```
1. MaitreyaToEnchoros.py - This scripts reads a text file with set of date,time and long/lat for a specific class of
2. asfer.cpp is executed with doClustering=true by which kNN and KMeans clustered data are obtained.
3. asferkmeansclustering.cpp and asferkNNclustering.cpp implement the KMeans and kNN respectively.
4. Set of strings from any of the clusters has to be written manually in asfer.enchoros.clustered
5. asferlongestcommonsubstring.cpp - Within each cluster a pairwise longest common substring is found out.
6. TranslateClusterPairwiseLCS.py - Translates the longest common substring to textual rule description mined from ab

-----------------------------------------------------------------------------------------
(FEATURE - DONE) 138. Commandline sequence for Mining Astronomical Datasets (e.g Earthquakes as above)
using KMeans and kNN - Commits as on 4 February 2015
-----------------------------------------------------------------------------------------
138.1 In asfer.cpp, set doClustering=true and build asfer binary
138.2 Set asfer.enchoros to asfer.enchoros.ascrelative or asfer.enchoros.zodiacal
138.3 $./asfer 2>&1 > logfile1
138.4 From logfile1 get maximum iteration clustered data for any cluster and update asfer.enchoros.clustered
138.5 In asfer.cpp set doLCS=true and build asfer binary
138.6 $./asfer 2>&1 > logfile2
138.7 grep "Longest Common Substring for" logfile2 2>&1 > python-src/asfer.ClusterPairwiseLCS.txt
138.8 sudo python TranslateClusterPairwiseLCS.py | grep "Textually" 2>&1 >> MinedRulesFromDatasets.earthquakes

-----------------------------------------------------------------------------------
(FEATURE - DONE) 139. BigData Analytics subsystem (related to point 64,65 on software analytics)
-----------------------------------------------------------------------------------
139.1 (DONE) As mentioned in commit notes(13February2015) below, new multipurpose Java Hadoop MapReduce
code has been added to a bigdata_analytics/ directory. At present it computes the frequencies of astronomical entitie

139.2 VIRGO Linux Kernel has following design choices to interface with the machine learning code
in AsFer :
        139.2.1 (DONE) the kernel module does an upcall to userspace asfer code - already this facility exists in VIR
        139.2.2 (INITIAL VIRGO COMMITS - DONE) the analytics subsystem creates a policy config file /etc/virgo_kernel
        139.2.3 (INITIAL ASFER COMMITS - DONE) kernel module implements the machine learning algorithm in C (C++ in k

Diagrams depicting the above options have been uploaded at: http://sourceforge.net/p/asfer/code/HEAD/tree/asfer-docs/


-----------------------------------------------------------------------------
(FEATURE - DONE) Commits as on 4 February 2015
-----------------------------------------------------------------------------
C++ implementation of Knuth-Morris-Pratt String Match Algorithm added to repository.

-----------------------------------------------------------------------------
(FEATURE - DONE) Commits as on 9 February 2015
-----------------------------------------------------------------------------
Python+R script for DFT analysis of multimedia data has been added to repository.

-----------------------------------------------------------------------------
(FEATURE - DONE) Commits as on 13 February 2015
-----------------------------------------------------------------------------
Initial commits for BigData Analytics (for mined astro datasets) using Hadoop 2.6.0 MapReduce:

- new folder bigdata_analytics has been added
- hadoop_mapreduce is subfolder of above
- A Hadoop Java MapReduce implementation to compute frequencies of astronomical entities in MinedRulesFromDatasets.ea
- This jar was executed on a Single Node Hadoop Cluster as per the documentation at:
        - http://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial
        - http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html
- The HDFS filesystem data is in hdfs_data - /user/root output with frequencies are in userroot_output and namenode a
- hdfs_data also has commandlines history and the console output logs for above.

---------------------------------------------------------------------
(FEATURE - DONE) Commits as on 17 February 2015
---------------------------------------------------------------------
Python implementations for LogLog and HyperLogLog cardinality estimation for streamed multiset data have been added t

---------------------------------------------------------------------
(FEATURE - DONE) Commits as on 18 February 2015
---------------------------------------------------------------------
Python implementations for CountMinSketch(frequencies and heavy-hitters) and BloomFilter(membership) for Streaming Da

---------------------------------------------------------------------
(FEATURE - DONE) Commits as on 19,20 February 2015
---------------------------------------------------------------------
Python clients for Apache Hive and HBase have been added to repository and invoked from
Stream Abstract Generator script as 2-phase streamer for Streaming_<algorithm> scripts.
Hive,Pig,HBase HDFS and script data added to repository in java-src/bigdata_analytics.

-----------------------------------------------------------------------------
(FEATURE - DONE) 140. Schematic for Apache Cassandra/Hive/HBase <=> AsFer Streaming_<algorithm> scripts interface
-----------------------------------------------------------------------------

($) BigData Source(e.g MovieLens) => Populated in Apache Cassandra, Apache HiveQL CLI(CREATE TABLE..., LOAD DATA LOCA
```

```
2407
2408  ($) Apache Hive or HBase SQL/NoSQL or Cassandra table => Read by Apache Hive or HBase or Cassandra python client => S
2409
2410  ----------------------------------------------------------------------
2411  (FEATURE - DONE) Commits as on 25 February 2015
2412  ----------------------------------------------------------------------
2413  Added the Hive Storage client code for Streaming Abstract Generator __iter__ overridden function. With this Streaming
2414
2415  ----------------------------------------------------------------------
2416  (DONE) 141. Classpaths required for Pig-to-Hive interface - for pig grunt shell in -x local mode
2417  ----------------------------------------------------------------------
2418  1. SLF4J jars
2419  2. DataNucleus JDO, core and RDBMS jars (has to be version compatible)
2420  in pig/lib and hive/lib directories:
2421  datanucleus-api-jdo-3.2.6.jar
2422  datanucleus-core-3.2.10.jar
2423  datanucleus-rdbms-3.2.9.jar
2424  3. Derby client and server jars
2425  4. Hive Shims jars
2426  hive-shims-0.11.0.jar
2427  5. All Hadoop core jars in /usr/local/hadoop/share/hadoop:
2428  common hdfs httpfs kms mapreduce tools yarn
2429  6. HADOOP_CONF_DIR($HADOOP_HOME/etc/hadoop) is also required in classpath.
2430
2431  ----------------------------------------------------------------------
2432  (DONE) 142. Classpaths required for Pig-to-HBase interface
2433  ----------------------------------------------------------------------
2434  In addition to the jars above, following cloudera trace jars are required:
2435  htrace-core-2.01.jar and htrace-1.45.jar
2436
2437  hadoop2_core_classpath.sh shell script in bigdata_analytics exports all the jars in (141) and (142).
2438
2439  ----------------------------------------------------------------------
2440  (FEATURE - DONE) Commits as on 26 February 2015
2441  ----------------------------------------------------------------------
2442  Pig-HBase stream_data table creation and population related Pig scripts, HDFS data and screenshots have been added to
2443
2444  ----------------------------------------------------------------------
2445  (FEATURE - DONE) Commits as on 27 February 2015
2446  ----------------------------------------------------------------------
2447  Cassandra Python Client has been added to repository and Streaming_AbstractGenerator.py has been
2448  updated to invoke Cassandra in addition to Hive,HBase and File storage. Cassandra data have been added
2449  in bigdata_analytics/
2450
2451  ----------------------------------------------------------------------
2452  143. (FEATURE - DONE) Storage Abstraction in AsFer - Architecture Diagram
2453  ----------------------------------------------------------------------
2454  Architecture diagram for Hive/Cassandra/HBase/File NoSQL and other storage abstraction implemented in python has been
2455
2456  ----------------------------------------------------------------------
2457  144. (FEATURE - DONE) Apache Spark and VIRGO Linux Kernel Analytics and Commits as on 6 March 2015
2458  ----------------------------------------------------------------------
2459  1. Prerequisite: Spark built with Hadoop 2.6.0 with maven commandline:
2460  mvn -Pyarn -Phadoop-2.4 -Dhadoop.version=2.6.0 -DskipTests package
2461
2462  2. Spark Python RDD MapReduce Transformation script for parsing the most frequent source IP address
2463  from Uncomplicated FireWall logs in /var/log/kern.log has been added to repository. This parsed
2464  IP address can be set as a config in VIRGO kernel_analytics module (/etc/virgo_kernel_analytics.conf)
2465
2466  ----------------------------------------------------------------------
2467  (FEATURE - DONE) Commits as on 10 March 2015
2468  ----------------------------------------------------------------------
2469  Spark python script updated for parsing /var/log/udev and logs in python-src/testlogs.
2470  (spark-submit Commandline: bin/spark-submit /media/shrinivaasanka/0fc4d8a2-1c74-42b8-8099-9ef78d8c8ea22/home/kashrini
2471
2472  ----------------------------------------------------------------------
2473  145. (FEATURE - DONE) Commits as on 26 March 2015
2474  ----------------------------------------------------------------------
2475  New python script to fetch stock quotes by ticker symbol that can be used for Streaming_<algorithm> scripts has been
2476
2477  ----------------------------------------------------------------------
2478  146. (FEATURE - DONE) Commits as on 2 April 2015
2479  ----------------------------------------------------------------------
2480  New Python script to get Twitter tweets stream data for a search query has been added.
2481
2482  ----------------------------------------------------------------------
2483  147. (FEATURE - DONE) Related to Item 3 - Sequence Mining Implementation - Commits as on 3 April 2015
2484  ----------------------------------------------------------------------
2485  Python class that implements Apriori GSP algorithm for mining frequent subsequences in ordered sequences
2486  dataset has been added to repository. Though exponential, together with Longest Common Subsequence, Apriori GSP gives
2487
```

```
148. An example Class Association Rule Learnt from Sequence Mining:
--------------------------------------------------------------
For length 6 most frequent subsequences in earthquake astronomical data from 1900 are: ('conjoinedplanets',frequency)
--------------------------------------------------------------------------------------------------------------------
[('0', 313), ('4', 313), ('8', 313), ('3', 313), ('7', 313), ('2', 313), ('6', 313), ('1', 313), ('5', 313), ('9', 31

-------------------
indices for planets:
-------------------
 * 0 - for unoccupied
 * 1 - Sun
 * 2 - Moon
 * 3 - Mars
 * 4 - Mercury
 * 5 - Jupiter
 * 6 - Venus
 * 7 - Saturn
 * 8 - Rahu
 * 9 - Ketu

Some inferences can be made from above:
-------------------------------------
By choosing the creamy layer of items with support > 30 (out of 313 historic events) - ~10%:
[('14', 121), ('46', 66), ('56', 51), ('13', 51), ('34', 42), ('45', 36), ('9a', 33), ('79', 31), ('24', 30), ('16',

Above implies that:
-------------------
Earthquakes occur most likely when, following happen - in descending order of frequencies:
- Sun+Mercury (very common)
- Mercury+Venus
- Jupiter+Venus
- Sun+Mars
- Mars+Mercury
- Mercury+Jupiter
- Ketu is in Ascendant
- Saturn+Ketu
- Moon+Mercury
- Sun+Venus

Machine Learnt pattern above strikingly coincides with some combinations in Brihat Samhita (Role of Mars, Nodes, and

Some recent major intensity earthquakes having above sequence mined astronomical conjunctions :
-------------------------------------------------------------------------------------------------
Sendai Earthquake - 11 March 2011 14:46 - Sun+Mars in Aquarius, Mercury+Jupiter in Pisces
Nepal Earthquake - 25 April 2015 11:56 - Sun+Mars+Mercury in Aries
Chile Earthquake - 16 September 2015 22:55 -  Sun+Mars+Jupiter in Leo
All three have Sun+Mars coincidentally.


---------------------------------------------------------------
(FEATURE - DONE) Commits as on 5 April 2015
---------------------------------------------------------------
Textual translation of Class Association Rules added to SequenceMining.py with logs.

---------------------------------------------------------------
(FEATURE - DONE) Commits as on 13 April 2015
---------------------------------------------------------------
Python implementation of :
   - Part-of-Speech tagging using Maximum Entropy Equation of Conditional Random Fields(CRF) and
   - Viterbi path computation in HMM-CRF for Named Entity Recognition has been added to repository.

---------------------------------------------------------------
(FEATURE - DONE) Commits as on 15 April 2015
---------------------------------------------------------------
Named Entity Recognition Python script updated with:
  - More PoS tags
  - Expanded HMM Viterbi probabilities matrix
  - Feature function with conditional probabilities on previously labelled word

---------------------------------------------------------------
(FEATURE - DONE) Commits as on 17 April 2015
---------------------------------------------------------------
Twitter Streaming python script updated with GetStreamFilter() generator object for streaming tweets data.

---------------------------------------------------------------
(FEATURE - DONE) 149. Commits as on 10 May 2015
---------------------------------------------------------------
A Graph Search NetworkX+MatPlotLib Visualizer for WordNet has been implemented in python and has been added to reposi

---------------------------------------------------------------
(FEATURE - DONE) Commits as on 12 May 2015
```

```
2569  ------------------------------------------------------------
2570  WordNet Visualizer script has been updated to take and render the WordNet subgraph computed by Recursive Gloss Overla
2571  - http://arxiv.org/abs/1006.4458
2572  - http://www.nist.gov/tac/publications/2010/participant.papers/CMI_IIT.proceedings.pdf
2573  - https://sites.google.com/site/kuja27/PresentationTAC2010.pdf?attredirects=0
2574
2575  ------------------------------------------------------------
2576  (FEATURE - DONE) 150. Commits as on 14 May 2015
2577  ------------------------------------------------------------
2578  Updated WordNet Visualizer with:
2579  - bidirectional edges
2580  - graph datastructure changed to dictionary mapping a node to a list of nodes (multigraph)
2581  - With these the connectivity has increased manifold as shown by gloss overlap
2582  - the graph nodes are simply words or first element of lemma names of the synset
2583  - more networkx drawing layout options have been added
2584
2585  Added code for:
2586  - removing self-loops in the WordNet subgraph
2587  - for computing core number of each node (maximum k such that node is part of k-core decomposition)
2588
2589  In the input example, nodes "India" and "China" have core numbers 15 and 13 respectively which readily classify the d
2590  class "India and China". Thus a new unsupervised classifier is obtained based on node degrees.
2591
2592  ------------------------------------------------------------
2593  (FEATURE - DONE) Commits as on 16 May 2015
2594  ------------------------------------------------------------
2595  - added sorting the core numbers descending and printing the top 10% core numbers which are prospective classes the d
2596  This is a new unsupervised text classification algorithm and is based on recursive gloss overlap algorithm in http://
2597  http://www.nist.gov/tac/publications/2010/participant.papers/CMI_IIT.proceedings.pdf. A major feature of this algorit
2598
2599  ------------------------------------------------------------
2600  (FEATURE - DONE) 151. Commits as on 18 May 2015
2601  ------------------------------------------------------------
2602  An interesting research was done on the wordnet subgraph obtained by Recursive Gloss Overlap algorithm:
2603  - PageRank computation for the WordNet subgraph was added from NetworkX library. This is probably one of the first ap
2604  a graph other than web link graph.
2605  - The objective was to find the most popular word in the subgraph obtained by the Random Walk Markov Model till it st
2606  - The resultant word with maximum pagerank remarkably coincides with the most probable class of the document and is a
2607  ranking to core numbers of nodes ranked descending.
2608  - Some random blog text was used.
2609  - Above was mentioned as a theoretical statement in 5.12 of http://www.nist.gov/tac/publications/2010/participant.pap
2610
2611  ------------------------------------------------------------
2612  (FEATURE - DONE) Commits as on 19 May 2015
2613  ------------------------------------------------------------
2614  A primitive text generation from the definition graph has been implemented by:
2615  - computing the k-core of the graph above certain degree k so that the core is dense
2616  - each edge is mapped to a relation - at present "has" , "is in" are the only 2 relations (more could be added based
2617  hypernyms)
2618  - Each edge is output as "X <relation> Y"
2619  - Above gives a nucleus text extracted from the original document
2620  - It implements the theory mentioned in: https://sites.google.com/site/kuja27/DocumentSummarization_using_SpectralGra
2621
2622  ------------------------------------------------------------
2623  (FEATURE - DONE) Commits as on 20 May 2015
2624  ------------------------------------------------------------
2625  Hypernyms/Hyponyms based sentence construction added to WordNet Visualizer. The core-number and pagerank based classi
2626
2627  ------------------------------------------------------------
2628  (FEATURE - DONE) Commits as on 21 May 2015
2629  ------------------------------------------------------------
2630  Updated the Visualizer to print the WordNet closure of hypernyms and hyponyms to generate a blown-up huge sentence. T
2631  uncovers new vertices and edges in the definition graph (strict supergraph of RGO graph) and it is a mathematically g
2632  (similar to first order logic and functional programming compositionality) and mimicks an ideal human-thinking proces
2633
2634  ------------------------------------------------------------
2635  (FEATURE - DONE) 151. Commits as on 22,23,24 May 2015
2636  ------------------------------------------------------------
2637  Added code for computing recursive composition of lambda functions over the RGO graph relations(edges):
2638  - For this the Depth First Search Tree of the graph is computed with NetworkX
2639  - The lambda function is created for each edge
2640  - Composition is done by concatenating a recursive parenthesisation string of lambda expressions:
2641          for each DFS edge:
2642                  Composed_at_tplus1 = Composed_at_t(lambda <edge>)
2643  - DFS is chosen as it mimicks a function invocation stack
2644  - Lambda composition closure on the Recursive Gloss Overlap graph has been rewritten to create a huge lambda composit
2645  not exist in WordNet parlance, because the Recursive Gloss OVerlap algorithm grows a graph recursively from WordNet S
2646  - Thus a tree is built which is made into a graph by pruning duplicate edges and word vertices. WordNet graph does no
2647  - Thus RGO in a way adds new hierarchical recursive relationships to WordNet based on Synset definition.
2648  - The lambda composition obtained above is a mathematical or lambda calculus representation of a text document - Natu
2649
```

```
------------------------------------------------------------
(FEATURE - DONE) Commits as on 26 May 2015
------------------------------------------------------------
Added initial code for Social Network Analyzer for Twitter following to create a NetworkX graph and render it with Ma

------------------------------------------------------------
(FEATURE - DONE) Commits as on 3 June 2015
------------------------------------------------------------
Bonacich Power Centrality computation added to Twitter Social Network Analyzer using PageRank computation.

------------------------------------------------------------
(FEATURE - DONE) Commits as on 4 June 2015
------------------------------------------------------------
- Eigen Vector Centrality added to Twitter Social Network Analyzer.
- File storage read replaced with Streaming_AbstractGenerator in Streaming_HyperLogLogCounter.py and Streaming_LogLog

------------------------------------------------------------
(FEATURE - DONE) 152. Commits as on 7 June 2015
------------------------------------------------------------
New Sentiment Analyzer script has been added to repository:
- This script adds to WordNet Visualizer with Sentiment analysis using SentiWordNet from NLTK corpus
- added a Simple Sentiment Analysis function that tokenizes the text and sums up positivity and negativity score
- A non-trivial Sentiment Analysis based on Recursive Gloss Overlap graph - selects top vertices with high core numbe
and elicits the positivity and negativity of those vertex words.
- Above is intuitive as cores of the graph are nuclei centering the document and the sentiments of the vertices of th
are important which decide the sentiment of the whole document.
- Analogy: Document is akin to an atom and the Recursive Gloss Overlap does a nuclear fission to extricate the inner
(subatomic particles are the vertices and forces are edges)
- Instead of core numbers, page_rank can also be applied (It is intriguing to see that classes obtained from core_num
and page_rank coincide to large extent) and it is not unusual to classify a document in more than one class (as it is

Above script can be used for any text including social media and can be invoked as a utility from SocialNetworkAnalys
Microblogging tweets are more crisp and "emotionally precise" i.e. extempore - convey instantaneous sentiment (withou

------------------------------------------------------------------
(FEATURE - DONE) Commits as on 8 June 2015
------------------------------------------------------------------
Commits for:
 - fixing errors due to NLTK API changes in lemma_names(), definition() ... [ variables made into function calls ]
 - Unicode errors
 - Above were probably either due to Ubuntu upgrade to 15.04 which might have added some unicode dependencies and/or
   (SentimentAnalyzer.py already has been updated to reflect above)

------------------------------------------------------------------
(FEATURE - DONE) Commits as on 10 June 2015
------------------------------------------------------------------
Sentiment Analysis for twitter followers' tweets texts added to SocialNetworkAnalysis_Twitter.py which invokes Sentim

------------------------------------------------------------------
(FEATURE - DONE) Commits as on 13 June 2015
------------------------------------------------------------------
Sentiment Analysis for tweet stream texts added to SocialNetworksAnalysis_Twitter.py which invokes SentimentAnalyzer.

------------------------------------------------------------------------------------------------------------------
153. (FEATURE - THEORY - Minimum Implementation DONE) Sentiment Analysis as lambda function composition of the Recurs
------------------------------------------------------------------------------------------------------------------
SentiWordNet based scoring of the tweets and texts have some false positives and negatives. But the lambda compositic
Recursive Gloss Overlap graph is done by depth-first-search - set of composition trees which overlap. If each edge is
a function of sentiment scores of two vertex words, then lambda composition performed over the graph is a better repr
of the document. This is at present a conjecture only. An implementation of this has been added to SentimentAnalyzer.
Propagation of a sentiment potential in the Recursive Gloss Overlap graph considering it as a Bayesian graphical mode

------------------------------------------------------------------------
Commits as on 15 June 2015
--------------------------
NeuronRain - (AsFer+USBmd+VIRGO+KingCobra) - version 15.6.15 release tagged
Most of features and bugfixes are in AsFer and VIRGO
------------------------------------------------------------------------

------------------------------------------------------------------------------------------------------------------
154. (FEATURE - DONE) Belief Propagation and RGO - Commits as on 20 June 2015
------------------------------------------------------------------------------------------------------------------
SentimentAnalyzer is updated with a belief propagation algorithm (Pearl) based Sentiment scoring by
considering Recursive Gloss Overlap Definition Graph as graphical model and sentiwordnet score for
edge vertices as the belief potential propagated in the bayesian network (RGO graph). Since the
sentiment is a collective belief extracted from a text rather than on isolated words and the
sentiwordnet scores are probabilities if normalized, Sentiment Analysis is reduced to Belief Propagation problem.
The previous Belief Propagation function is invoked from SocialNetworkAnalysis_<> code to Sentiment Analyze tweet str

------------------------------------------------------------------------------------------------------------------
155. Updates to https://sites.google.com/site/kuja27/DocumentSummarization_using_SpectralGraphTheory_RGOGraph_2014.pd
```

```
2731 ------------------------------------------------------------------------------------------------------------------------
2732 Algorithms and features based on Recursive Gloss Overlap graph:
2733 155.1 RGO visualizer
2734 155.2 RGO based Sentiment Analyzer - Belief Propagation in RGO as a graphical model
2735 155.3 Graph Search in text - map a text to wordnet relations by RGO graph growth - prints the graph edges which are r
2736 a text
2737 155.4 New unsupervised text classifier based in RGO core numbers and PageRank
2738 155.5 Social Network Analyzer (Tweets analysis by RGO graph growth)
2739 155.5 Lambda expression construction from the RGO graph
2740 155.6 Sentence construction by composition of edges
2741
2742 ------------------------------------------------------------------------
2743 156. (FEATURE - DONE) Commits as on 4 July 2015 - RGO sentiment analyzer on cynical tweets
2744 ------------------------------------------------------------------------
2745 Added query for a "elections" to stream tweets related to it. Few sample cynical tweets were sentiment-analyzed with
2746 RGO Belief Propagation SentimentAnalyzer that gives negative sentiment scores as expected whereas trivial SentiWordNe
2747 gives a positive score wrongly.
2748
2749 ------------------------------------------------------------------------------------------------------------------------
2750 157. (THEORY) Recursive Gloss Overlap graphical model as a Deep Learning algorithm that is an alternative to Perceptr
2751 ------------------------------------------------------------------------------------------------------------------------
2752 Point 155 mentions the diverse applications of Recursive Gloss Overlap algorithm each of which touch upon multiple fa
2753
2754 ------------------------------------------------------------------------------------------------------------------------
2755 158. (THEORY) Graph Discovery (or) Graph Guessing and EventNet (related to EventNet points 70-79)
2756 ------------------------------------------------------------------------------------------------------------------------
2757 EventNet mentioned previously is an infinite cause-effect graph of event vertices with partakers for each event
2758 (kind of a PetriNet yet different). A special case of interest is when only few subgraphs of a giant EventNet is avai
2759 often necessary to discover or guess the complete graph of causality with partakers. A familiar example is a crime sc
2760 where:
2761  - it is necessary to recreate the complete set of events and their causalities with partakers of a crime (reverse-en
2762  - but only few clues or none are available - which are akin to very sparse subgraphs of the crime scene EventNet
2763  - Guessing (100-x)% of the graph from x% clue subgraphs can be conjectured to be an NP problem - because non-determi
2764 EventNet path can be guessed. As a counting problem this is #P-complete (set of all paths among known subgraphs of ar
2765
2766 Another example: set of blind men try to make out an object by touch.
2767
2768 ------------------------------------------------------------------------------------------------------------------------
2769 159. (THEORY) Pattern Grammar and Topological Homeomorphism of Writing Deformations
2770 ------------------------------------------------------------------------------------------------------------------------
2771 Grammar for patterns like texts, pictures similar to RE,CFG etc., for pattern recognition:
2772    - example text grammar:  <a> := <o> <operator> <)>
2773    - example text grammar:  <d> := <o> <operator> <l>
2774    - example text grammar:  <v> := <\> <operator> </>
2775 Previous grammar is extensible to any topological shapes. For example, handwriting of 2 individuals are
2776 homeomorphic deformations.
2777
2778 References:
2779 -----------
2780 159.1 Topology and Graphics - https://books.google.co.in/books?id=vCc8DQAAQBAJ&pg=PA263&lpg=PA263&dq=homeomorphism+ar
2781
2782 ------------------------------------------------------------------------------------------------------------------------
2783 160. (THEORY) Cognitive Element
2784 ------------------------------------------------------------------------------------------------------------------------
2785 An element of a list or group is cognitive with respect to an operator if it "knows" about all or subset of other ele
2786    - In list 2,3,5,7,17 - 17 knows about 2,3,5,7 in the sense that 17 = 2+3+5+7 with + operator
2787    - In list "addon","add","on" - "addon" knows about "add" and "on" with concatenation operator
2788    - In list "2,3,6,8,9,10,..." - 6 knows about 2 and 3, 10 knows about 2 and 5 with factorization as operator.
2789 This might have an equivalent notion in algebra. Motivation for this is to abstract cognition as group operator. Esse
2790 to an equivalence relation graph where elements are related by a cognitive operator edge.
2791
2792
2793 ------------------------------------------------------------------------------------------------------------------------
2794 161. (THEORY) Philosophical intuition for P(Good) summation - Continuation of (129) and other previous related points
2795 ------------------------------------------------------------------------------------------------------------------------
2796 The apparent paradox in Perfect Voting can be reconciled as below to some extent:
2797 - Voting is done in parallel in real-world elections and not serially
2798 - Votes are counted in parallel - iterated integer addition in NC1 (http://www.ccs.neu.edu/home/viola/classes/gems-08
2799 - Thus if RHS is just Circuit-Value Problem and not a Circuit-SAT then LHS getting equated to RHS is not unusual - bc
2800 - Separation of RHS from LHS happens only when the Circuit-SAT is required in RHS.
2801 - P(Good) convergence in perfect voting implies that Voter SAT is not necessary if there is perfection (proving the c
2802 true even though the PRG is imperfect that operates to choose on a perfect set.
2803 - Parallelism implies NC circuits
2804 - Above applies to infinite majority also (Mark Fey)
2805 - RHS in worst-case can be EXP-Complete if the Majority+Voter SAT oracle circuit is of unrestricted depth and lot of
2806
2807 ------------------------------------------------------------------------------------------------------------------------
2808 162. (THEORY) Majority Voting Circuit with Boolean Function Oracles
2809 ------------------------------------------------------------------------------------------------------------------------
2810 If each voter has a boolean function to decide which has fanout > 1 gates instead of a formula, RHS Majority Voting b
2811 Decision tree, Certificate,  and other complexity measures automatically come into reckoning. Thus RHS circuit is an
```

```
References:
-----------
162.1 http://www.cs.cmu.edu/~odonnell/papers/barbados-aobf-lecture-notes.pdf
162.1 http://www.math.u-szeged.hu/~hajnal/research/papers/dec_surv.gz

-----------------------------------------------------------------------------------------------------
163. (THEORY) Parity and Constant Depth - intuition (not a formal proof, might have errors)
-----------------------------------------------------------------------------------------------------
Inductive Hypothesis:
---------------------
For depth d and n variables parity has n^k sized circuit.

For n+1 variables:
------------------
       XOR gate
       /      \
n-variable   (n+1)th variable
circuit

Each XOR gate is of atleast depth 2 or 3 with formula - (x /\ not y) V (not x /\ y). Thus for each additional variabl

-----------------------------------------------------------------------------------------------------
164. (FEATURE - DONE) Commits as on 16 September 2015
-----------------------------------------------------------------------------------------------------
cpp-src/cloud_move - Implementation of Move Semantics for Cloud objects:
-------------------------------------------------------------------------
This expands on the usual move semantics in C++ and implements a Perfect Forwarding of objects over cloud. A move cli

VIRGO kernel sockets code has been carried over and changed for userspace that uses traditional sockaddr_in and htons

C++ sockets reference code adapted for std::move - for use_addrinfo clause:
- getaddrinfo linux man pages
- http://codebase.eu/tutorial/linux-socket-programming-c/ (addrinfo instead of usual sockaddr_in and htons)

Move semantics schematic:
-------------------------
                                  &
 member fn temp arg lvalue <-------source data rvalue
                   |                  /
               &   |                 / && (removes a temp copy)
                   |                /
                   V               /
client proxy destination lvalue<-/
                   |
                   |-------------------------> cloud server destination

lvalue reference& does additional copy which is removed by rvalue reference&& to get the rvalue directly. Move client

-----------------------------------------------------------------------------------------------
165. (FEATURE - DONE) Commits as on 18 September 2015
-----------------------------------------------------------------------------------------------
Cloud Perfect Forwarding - Google Protocol Buffer Currency :
-----------------------------------------------------------
Currency object has been implemented with Google Protocol Buffers - in cloud_move/protocol_buffers/ src_dir and out_d
- lack of JSON format language specific compilers
- XML is too complicated
- Protocol Buffers also have object serialization-to-text member functions in generated C++ classes.

Protocol Buffer compilation after change to currency object:
protoc -I=src_dir/ --cpp_out=out_dir/ src_dir/currency.proto

-----------------------------------------------------------------------------------------------
166. (THEORY) Related to 65 - Debug Analytics (as part of Software Analytics)
-----------------------------------------------------------------------------------------------
Debugging software is painful process with repetitive labour - For example kernel and device driver development has t
1. Writing the kernel patch code for some deep kernel panics.
2. Kernel incremental or full build.
3. Test the patch.
(1),(2) and (3) are sometimes frustratingly repetitive. If the debugging is represented as a state machine automaton,

References:
166.1 Learning Finite State Machines - https://www.cs.upc.edu/~bballe/other/phdthesis.pdf

-----------------------------------------------------------------------------------------------
167. (THEORY) Prestige based ranking and Condorcet Elections
-----------------------------------------------------------------------------------------------
Web Search Engine Rankings by prestige measures are n-tuples of rankings (for n candidate web pages where n could be

-------------------------------------------------------------------------
168. (FEATURE - DONE) Commits as on 20,21,22 October 2015
```

```
2893 ------------------------------------------------------------------------
2894 Initial code for LinkedIn crawl-scrape. Uses Python-linkedin from https://github.com/ozgur/python-linkedin with
2895 some additional parsing for url rewriting and wait for authurl input:
2896 - prints a linkedin url which is manually accessed through curl or browser
2897 - created redirect_url with code is supplied to raw_input "authurl:"
2898 - parses the authcode and logs-in to linkedin to retrieve profile data; get_connections() creates a forbidden error
2899
2900 Logs for the above has been added to testlogs
2901
2902 ------------------------------------------------------------------------
2903 169. (FEATURE - DONE) Commits as on 28 October 2015
2904 ------------------------------------------------------------------------
2905 Deep Learning Convolution Perceptron Python Implementation.
2906
2907 ------------------------------------------------------------------------
2908 170. (FEATURE - DONE) Commits as on 29 October 2015, 1 November 2015
2909 ------------------------------------------------------------------------
2910 Deep Learning BackPropagation Multilayered Perceptron Python Implementation.
2911
2912 ------------------------------------------------------------------------
2913 171. Commits as on 3 November 2015
2914 ------------------------------------------------------------------------
2915 DeepLearning BackPropagation implementation revamped with some hardcoded data removal.
2916
2917 ----------------------------------------------------------------------------------------
2918 172. (FEATURE - DONE) Hurricane Datasets Sequence Mining - Commits as on 4 November 2015
2919 ----------------------------------------------------------------------------------------
2920 Miscellaneous code changes:
2921 --------------------------
2922 - Weight updates in each iteration are printed in DeepLearning_BackPropagation.py
2923 - Changed maitreya_textclient path for Maitreya's Dreams 7.0 text client; Updated to read HURDAT2 NOAA Hurricane Data
2924 - Maximum sequence length set to 7 for mining HURDAT2 asfer.enchoros.seqmining
2925 - New files asfer.enchoros.ascrelative.hurricanes, asfer.enchoros.zodiacal.hurricanes have been added which are creat
2926 - an example chartsummary for Maitreya's Dreams 7.0 has been updated
2927 - 2 logs for SequenceMining for HURDAT2 encoded datasets and Text Class Association Rules learnt by SequenceMining.py
2928 - Increased number of iterations in BackPropagation to 100000; logs for this with weights printed are added in testlo
2929
2930 ================================================
2931 Sorted Candidate support for all subsequence lengths - gives an approximate pattern in dataset:
2932 [('0', 1333), ('4', 1333), ('8', 1333), ('3', 1333), ('7', 1333), ('2', 1333), ('6', 1333), ('1', 1333), ('5', 1333),
2933 size of dataset =  1333
2934
2935 ========================================
2936 Sun , Mercury ,
2937 ========================================
2938 Class Association Rule  12  mined from dataset: with frequencies: 23.6309077269  percentage
2939 ========================================
2940 Mercury , Venus ,
2941 ========================================
2942 Class Association Rule  13  mined from dataset: with frequencies: 19.5798949737  percentage
2943 ========================================
2944 Venus , Saturn ,
2945 ========================================
2946 Class Association Rule  14  mined from dataset: with frequencies: 14.2535633908  percentage
2947 ========================================
2948 Saturn , Rahu ,
2949 ========================================
2950 Class Association Rule  15  mined from dataset: with frequencies: 14.1785446362  percentage
2951 ========================================
2952 Jupiter , Ketu ,
2953 ========================================
2954 Class Association Rule  16  mined from dataset: with frequencies: 13.9534883721  percentage
2955 ========================================
2956 Mercury , Ketu ,
2957 ========================================
2958 Class Association Rule  17  mined from dataset: with frequencies: 13.503375844  percentage
2959 ========================================
2960 Mars , Mercury ,
2961 ========================================
2962 Class Association Rule  18  mined from dataset: with frequencies: 12.3780945236  percentage
2963 ========================================
2964 Jupiter , Saturn ,
2965 ========================================
2966 Class Association Rule  19  mined from dataset: with frequencies: 12.0780195049  percentage
2967 ========================================
2968 Mercury , Jupiter ,
2969 ========================================
2970 Class Association Rule  20  mined from dataset: with frequencies: 11.9279819955  percentage
2971 ========================================
2972 Sun , Mars ,
2973 ========================================
```

```
Class Association Rule  21  mined from dataset: with frequencies: 11.8529632408  percentage
========================================
Jupiter , Venus ,
========================================
Class Association Rule  22  mined from dataset: with frequencies: 11.0277569392  percentage
========================================
Jupiter , Rahu ,
========================================
Class Association Rule  23  mined from dataset: with frequencies: 10.0525131283  percentage
========================================
Sun , Moon ,
========================================
Class Association Rule  24  mined from dataset: with frequencies: 9.60240060015  percentage
========================================
Moon , Mars ,
========================================
Class Association Rule  25  mined from dataset: with frequencies: 8.70217554389  percentage
========================================
Sun , Mercury , Venus ,
========================================
Class Association Rule  26  mined from dataset: with frequencies: 8.6271567892  percentage
========================================
Mars , Venus ,
========================================
Class Association Rule  27  mined from dataset: with frequencies: 8.47711927982  percentage
========================================
Rahu , Ascendant ,
========================================
Class Association Rule  28  mined from dataset: with frequencies: 8.10202550638  percentage
========================================
Moon , Jupiter ,
========================================
Class Association Rule  29  mined from dataset: with frequencies: 7.951987997  percentage
========================================
Ketu , Ascendant ,
========================================
Class Association Rule  30  mined from dataset: with frequencies: 7.72693173293  percentage
========================================
Mars , Mercury , Venus ,
========================================

Above are excerpts from the logs added to testlogs/ for Sequence mined from HURDAT2 dataset for hurricanes from 1851

================================================================================================================
173. (FEATURE - DONE) BigData Storage Backend Abstraction subsystem for AsFer - Commits as on 5 November 2015 :
----------------------------------------------------------------------------------------------------------------
- These are initial minimal commits for abstracting storage of ingested bigdata for AsFer Machine Learning code
- A dependency injection implementation of database-provider-specific objects is added to repository.
- Presently MySQLdb based MySQL backend has been implemented with decoupled configuration which are injected into Abs

Schematic Dependency Injected Object Graph:
-------------------------------------------
                                      Injector ==============================================================
                                                                                                             |
                                                                                                             |
                                                                                                             V
               MySQL_Configuration ---injects config ----> MySQL_DBBackend ---- injects connection ----> Abstract_DB
                                                                                                            /\
                                                                                                             |
               xxx_Configuration ---injects config ----> xxx_DBBackend ---- injects connection -----------|

----------------------------------------------------------------------------------------------------------------
174. (FEATURE - DONE) AsFer C++ - Python Integration - Embedding Python in AsFer C++ - Commits as on 11 November 2015
----------------------------------------------------------------------------------------------------------------
Python Embedding in C++ implementation has been added and invoked from asfer main entrypoint with a boolean flag. Thi

Schematic Diagram:
------------------
        AsFer C++ -----> C Python Embedding ------> AsFer Python  --------> Machine Learning algorithms on Spark RDD
        /\                                                                                                      V
         |                                                                                                      |
         --------------------<--------- userspace upcall ---------------VIRGO kernel analytics and other kernel modul

----------------------------------------------------------------------------------------------------------------
175. (FEATURE - DONE) Config File Support for AsFer C++ and Python Machine Learning Algorithms - Commits as on 12 Nov
----------------------------------------------------------------------------------------------------------------
Config File support for asfer has been added. In asfer.cpp main(), read_asfer_config() is invoked which reads config

--------------------------------------------------------------------------------------------------------------
176. (THEORY) Isomorphism of two Document Definition Graphs
--------------------------------------------------------------------------------------------------------------
```

```
http://arxiv.org/abs/1006.4458 and http://www.nist.gov/tac/publications/2010/participant.papers/CMI_IIT.proceedings.p

------------------------------------------------------------------------------------------
177. Commits as on 13 November 2015
------------------------------------------------------------------------------------------
    - Corrections to Convolution Map Neuron computation with per-coordinate weight added
    - Removed hardcoded convolution stride and pooling width in the class
    - testlogs following input bitmap features have been added with the output of 5 neurons in final layer:
        - Without Zero inscribed
        - With Zero inscribed in bitmap as 1s
        - With Bigger Zero inscribed in bitmap as 1s
        - With Biggest Zero inscribed in bitmap as 1s
      The variation of final neural outputs can be gleaned as the size of the pattern increases from none to biggest.
      The threshold has to be set to neural output without patterns. Anything above it shows a pattern.
    - Added a config variable for invoking cloud perfect forwarding binaries in KingCobra in asfer.conf
    - config_map updated in asfer.cpp for enableCloudPerfectForwarding config variable

------------------------------------------------------------------------------------------
178. (THEORY) Approximate Machine Translation with Recursive Gloss Overlap Definition Graph
------------------------------------------------------------------------------------------
For natural language X (=English) an RGO graph can be constructed (based on algorithms in http://arxiv.org/abs/1006.4

------------------------------------------------------------------------------------------------------
179. (FEATURE - DONE) Web Spider Implementation with Scrapy Framework - Commits as on 16 November 2015
------------------------------------------------------------------------------------------------------
Initial commits for a Web Spider - Crawl and Scrape - done with Scrapy framework. Presently it crawls Google News for
crawled items in items.py

------------------------------------------------------------------------------------------------------
180. (FEATURE - DONE) Sentiment Analyzer Implementation for Spidered Texts - Commits as on 16 November 2015
------------------------------------------------------------------------------------------------------
Sentiment Analyzer implementation for Crawled-Scraped Google News Search result texts with testlogs and screenshots c
been added. This has been made a specialized analyzer for spidered texts different from twitter analyzer.

------------------------------------------------------------------------------------------------------
181. Commits as on 17 November 2015
------------------------------------------------------------------------------------------------------
Requirements.txt for package dependencies have been added to asfer-docs/.
WebSpider Crawl-Scraped URLs descriptions are inserted into MySQL table (asfer_webspider) by importing backend/Abstra
implementation. Logs for this Scrapy crawl has been added to webspider/testlogs. WebSpider.py parse() has been update

------------------------------------------------------------------------------------------------------
182. Commits as on 19 November 2015
------------------------------------------------------------------------------------------------------
- Added more stop words in spidered text and commented text generation in SocialNetworkAnalysis_WebSpider.py
- logs and screenshots for this has been added to testlogs/
- crawl urls in WebSpider.py has been updated
- Added more edges in definition graph by enumerating all the lemma names of a keyword in previous iteration. This ma
and probability of a node having more k-core number increases and classification is more accurate.
- logs and screenshots for updated web spidered news texts of two topics "Theoretical Computer Science" and "Chennai"

------------------------------------------------------------------------------------------------------
183. Commits as on 20 November 2015
------------------------------------------------------------------------------------------------------
- Updated Spidered text for Sentiment Analysis
- Changed the Sentiment Analysis scoring belief potential propagation algorithm:
  - Two ways of scoring have been added - one is based on DFS tree of k-core subgraph of the larger wordnet subgraph
  - Sentiment thus is computed only for the core of the graph which is more relevant to the crux or class of the docu
vertices are ignored.
  - the fraction score is multiplied by a heuristic factor to circumvent floating points

------------------------------------------------------------------------------------------------------
184. Commits as on 23 November 2015
------------------------------------------------------------------------------------------------------
- Updated SentimentAnalyzer.py and SocialNetworkAnalysis_Twitter.py scripts with core number and k-core DFS belief pc
similar to SocialNetworkAnalysis_WebSpider.py.
- logs and screenshots for twitter followers graph and tweet RGO graph Sentiment Analysis have been added.
- Example tweet is correctly automatically classified into "Education" class based on top core number of the vertices
Sentiment Analysis of the tweet:
- K-Core DFS belief_propagated_posscore: 244.140625
  K-Core DFS belief_propagated_negscore: 1.0
- Core Number belief_propagated_posscore: 419095.158577
  Core Number belief_propagated_negscore: 22888.1835938
- Above Sentiment Analysis rates the tweet with positive tonality.

------------------------------------------------------------------------------------------------------
185. Commits as on 25 November 2015
------------------------------------------------------------------------------------------------------
- New WordNetPath.py file has been added to compute the path in a WordNet graph between two words
- WordNetSearchAndVisualizer.py text generation code has been updated for importing and invoking path_between() funct
by which set of sentences are created from the RGO WordNet subgraph. This path is obtained from common hypernyms for
```

```
that enhances WordNet relations. Each edge in RGO graph is made a hyperedge due to this hypernym path finding.
- logs and screenshots have been added for above

The text generated for the hypernym paths in WordNet subgraph in testlogs/ is quite primitive and sentences are conne

---------------------------------------------------------------------------------------------------------
186. Commits as on 4 December 2015
---------------------------------------------------------------------------------------------------------
All the Streaming_<>.py Streaming Algorithm implementations have been updated with:
- hashlib ripemd160 hash MD algorithm for hash functions and return hexdigest()
- USBWWAN byte stream data from USBmd print_buffer() logs has been added as a Data Storage and Data Source
- logs for the above have been added to testlogs/
- Streaming Abstract Generator has been updated with USB stream data iterable and parametrized for data source and st
- Some corrections to the scripts

---------------------------------------------------------------------------------------------------------
187. Commits as on 7 December 2015
---------------------------------------------------------------------------------------------------------
- USB stream file storage name updated in Streaming_AbstractGenerator
- Corrections to CountMinSketch - hashing updated to include rows (now the element frquencies are estimated almost ex
- logs for above updated to CountMinSketch
Added Cardinality estimation with LogLog and HyperLogLog counters for USB stream datasets
- HyperLogLog estimation: ~110 elements
- LogLog estimation: ~140 elements

---------------------------------------------------------------------------------------------------------
188. Commits as on 8 December 2015
---------------------------------------------------------------------------------------------------------
- Updated the Streaming LogLogCounter and HyperLogLogCounter scripts to accept StreamData.txt dataset from Abstract G
Spark MapReducer script for Streaming Data sets for comparison of exact data with Streaming_<algorithm>.py estimation
- Added logs for the Counters and Spark MapReducer script on the StreamData.txt
- LogLog estimation: ~133
- HyperLogLog estimation: ~106
- Exact cardinality: 104

---------------------------------------------------------------------------------------------------------
189. (FEATURE - DONE) Commits as on 9 December 2015
---------------------------------------------------------------------------------------------------------
- New python implementation for CountMeanMinSketch Streaming Data Frequency Estimation has been added which is an imp
CountMinSketch that computes median of average of estimator rows in sketch
- Corrections to CountMinSketch hashing algorithms and Sketch width-depth as per the error bounds has been made
- Spark MapReducer prints the number of elements in the stream data set
- Logs for the above have been added to testlogs

---------------------------------------------------------------------------------------------------------
190. (FEATURE - DONE) Commits as on 11 December 2015
---------------------------------------------------------------------------------------------------------
Dependency Injection code commits for MongoDB backend - With this MongoDB is also a storage backend for AsFer algorit
- Abstract_DBBackend.py has been updated for both MySQL and MongoDB injections
- MongoDB configuration and backend connect/query code has been added. Backend is either populated by Robomongo or py
Streaming Abstract Generator iterable framework.
- With this AsFer supports both SQL(MySQL) and NoSQL(file,hive,hbase,cassandra backends in Streaming Abstract Generat
- log with a simple NoSQL table with StreamingData.txt and USBWWAN data has been added to testlogs/.
- MongoDB configuration has a database(asfer-database) and a collection(asfer-collection).
- MongoDB_DBBackend @provides pymongo.collection.Collection which is @inject-ed to Abstract_DBBackend
- Abstract_DBBackend changes have been reflected in Scrapy Web Spider - backend added as argument in execute_query()
- Abstract_DBBackend.py has a subtle problem:
        - Multiple @inject(s) are not supported in Python Injector
        - only the innermost @inject works and outer @inject throws a __init__ argument errors in webspider/
        - Conditional @inject depending on backend is required but at present switching the order of @inject(s) circu
- most recent scrapy crawl logs for this have been added to webspider/testlogs

---------------------------------------------------------------------------------------------------------
191. (THEORY) An update and additions to runtime analysis of Recursive Gloss Overlap Algorithm in TAC 2010
(http://www.nist.gov/tac/publications/2010/participant.papers/CMI_IIT.proceedings.pdf)
---------------------------------------------------------------------------------------------------------
Analysis of runtime in the link above is too cryptic. An updated analysis of steps are given here:
- Nodes at topmost level are keywords in document. For each subsequent recursion step following analysis is per keywo
- Nodes at level i-1 are computed (base case) : x^(i-1) where x is average size of gloss definition
- Naive pairwise comparison of overlap is done at level i-1 which makes it x^2(i-1)
- Tree isomorphism algorithms can be optionally applied to reduce number of nodes getting recomputed. There are
polytime algorithms for subtree isomorphisms (www.cs.bgu.ac.il/~dekelts/publications/subtree.pdf)
- Nodes at level i-1 are reduced by Overlap(i-1) : x^(i-1) - Overlap(i-1)
- Maximum number Nodes at level i - are from gloss expansion of those at i-1 : (x^(i-1) - Overlap(i-1))*x
- Thus total time at level i is: Time_for_pairwise_comparison_to_find_gloss_verlap + Time_for_removing_isomorphic_noc
T(i) = sigma([x^(i-1) - Overlap(i-1)]^2 + subtree_isomorphism(i)))
- Above is naively upperbounded to O(x^(2d)) [as subtree isomorphism is polynomial in supertree and subtree vertices
- If number of vertices in RGO multipartite graph (ignoring isomorphism) constructed above is V=O(x^(d)), runtime is
far less than O(E*V^2) mentioned in TAC 2010 link because number of keywords in toplevel are less than number of edge
- On a related note runtime for intrinsic merit ranking of the RGO wordnet subgraph can not be equated per-se to rank
- Parallel Recursive Gloss Overlap graph construction on a cloud could reduce the runtime to O(W*V^2/c) where c is si
```

```
3217
3218  ----------------------------------------------------------------------------------------------------------------
3219  192. (FEATURE - DONE) Commits as on 14 December 2015
3220  ----------------------------------------------------------------------------------------------------------------
3221  - New Interview Algorithm script with NetworkX+Matplotlib rendering and takes as input a randomly crawled HTML webpag
3222  rip off script and style tags and write only text in the HTML page) has been added
3223  - Above script also computes the graph theoretic connectivity of the RGO wordnet subgraph based on Menger's theorem -
3224  required to disconnect the graph or equivalently number of node indpendent paths
3225  - logs and screenshots for the above have been added
3226  - WebSpider.py has been updated to crawl based on a crawling target parameter - Either a streaming website(twitter, s
3227  HTML webpage
3228
3229  ----------------------------------------------------------------------------------------------------------------
3230  193. (FEATURE - DONE) RGO graph complexity measures for intrinsic merit of a text document - Commits as on 15 Decembe
3231  ----------------------------------------------------------------------------------------------------------------
3232  - Interview Algorithm Crawl-Visual script has been updated with a DOT graph file writing which creates a .dot file fo
3233  - Also variety of connectivity and graph complexity measures have been added
3234  - Importantly a new Tree Width computing script has been added. NetworkX does not have API for tree width, hence a na
3235  through all subgraphs and finds intersecting subgraphs to connect them and form a junction tree, has been written. Th
3236
3237  ----------------------------------------------------------------------------------------------------------------
3238  194. Commits as on 16 December 2015
3239  ----------------------------------------------------------------------------------------------------------------
3240  - TreeWidth implementation has been corrected to take as input set of edges of the RGO graph
3241  - TreeWidth for set of subgraphs less than an input parameter size is computed as TreeWidth computation is exponentia
3242  and there are MemoryErrors in python for huge set of all subgraphs. For example even a small graph with 10 nodes give
3243  possible subgraphs
3244  - Spidered text has been updated to create a small RGO graph
3245  - Logs and screenshots have been added
3246  - Each subgraph is hashed to create a unique string for each subgraph
3247  - TreeWidth is printed by finding maximum set in junction tree
3248
3249  ----------------------------------------------------------------------------------------------------------------
3250  195. (THEORY) WordNet, Evocation, Neural networks, Recursive Gloss Overlap and Circuit Complexity
3251  ----------------------------------------------------------------------------------------------------------------
3252  WordNet and Evocation WordNet are machine learning models based on findings from psychological experiments. Are WordN
3253  related? Evocation WordNet which is network of words based on how evocative a word is of another word readily transla
3254  This is because a machine learning abstraction of neuron - perceptron - takes weights, inputs and biases and if the l
3255
3256  ----------------------------------------------------------------------------------------------------------------
3257  196. (FEATURE - DONE) Mined Rule Search in Astronomical Data - Commits as on 17 December 2015
3258  ----------------------------------------------------------------------------------------------------------------
3259  - New Mined Class Association Rule Search script has been added. This script searches the astronomical data with pars
3260  data with date, time and longitude-latitude queries.
3261  - Where this is useful is after mining the astronomical data with SequenceMining, there is a necessity to search when
3262  - Logs for this has been added in testlogs/ and chartsummary.rulesearch
3263
3264  ----------------------------------------------------------------------------------------------------------------
3265  197. Commits as on 18 December 2015
3266  ----------------------------------------------------------------------------------------------------------------
3267  - Interview Algorithm crawl-visual script changed for treewidth invocation
3268  - Spidered text changed
3269  - Rule Search script corrected to use datetime objects
3270  - Rule Search script corrected to use timedelta for incrementing date and time
3271  - logs and screenshots for above
3272  - Junction Tree for RGO graph - logs and screenshots
3273
3274  ----------------------------------------------------------------------------------------------------------------
3275  198. (THEORY) Star Complexity of Graphs - Complement Function circuit and Unsupervised Classification with RGO graph
3276  ----------------------------------------------------------------------------------------------------------------
3277  Star complexity of a graph defined in [Stasys Jukna] - http://www.thi.informatik.uni-frankfurt.de/~jukna/ftp/graph-co
3278   is the minimum number of union and intersection operations of star subgraphs required to create the larger graph whi
3279
3280  ----------------------------------------------------------------------------------------------------------------
3281  199. (FEATURE - DONE) Tornado Webserver, REST API and GUI for NeuronRain - Commits as on 22 December 2015
3282  ----------------------------------------------------------------------------------------------------------------
3283  Commits for NeuronRain WebServer-RESTfulAPI-GUI based on tornado in python-src/webserver_rest_ui/:
3284  - NeuronRain entrypoint based on tornado that reads a template and implements GET and POST methods
3285  - templates/ contains renderable html templates
3286  - testlogs/ has neuronrain GUI logs
3287  - RESTful API entrypoint <host:33333>/neuronrain
3288
3289  ----------------------------------------------------------------------------------------------------------------
3290  200. (FEATURE - DONE) NeuronRain as SaaS and PaaS - for commits above in (199)
3291  ----------------------------------------------------------------------------------------------------------------
3292  RESTful and python tornado based Graphical User Interface entrypoint that reads from various html templates and passe
3293  incoming concurrent requests to NeuronRain subsystems - AsFer, VIRGO, KingCobra, USBmd and Acadpdrafts - has bee adde
3294  simplest possible POST form without too much rendering (might require flask, twisted, jinja2 etc.,) for AsFer algorit
3295  This exposes a RESTful API for commandline clients like cURL. For example a cURL POST is done to NeuronRain as:
3296  cURL POST: curl -H "Content-Type: text/plain" -X POST -d '{"component":"AsFer","script":"<script_name>","arguments":"
3297  With this NeuronRain is Software-As-A-Service (SaaS) Platform deployable on VIRGO linux kernel cloud, cloud OSes and
```

```
More so, it is Platform-As-A-Service (PaaS) when run on a VIRGO cloud.

------------------------------------------------------------------------------------------------------------------------
201. (FEATURE - DONE) Apache Spark MapReduce Parallel Computation of Interview Algorithm Recursive Gloss Overlap Grap
 - Commits as on 24 December 2015
------------------------------------------------------------------------------------------------------------------------
- 2 python files for Spark MapReduce of Recursive Gloss Overlap graph construction has been added to repository
- These two implement Interview Algorithm Recursive Gloss Overlap graph construction and Map-Reduce functions that pa
each recursion step computing the gloss tokens from previous recursion level. This is the most important part of the
consumes lot of time in serial version along with overlap computation. Apache Spark has been recently gaining importa
- In this implementation, Resilient Distributed Dataset is the set of tokens in each recursion level and is parallell
huge Spark cluster.
- For example if Spark cluster has 10000 nodes, and each level of recursion produces X number of gloss tokens for gra
complexity in Spark is O(X/10000) where as serial version is O(X) with negligible network overhead. Spark has in-memc
minimizes network latency because of disk access.
- There were lot of implementation issues to make the parallellism. Map and Reduce functions use namedtuples to retur
field in namedtuple, there are internal pickling issues in Py4J - Python4Java which PySpark internally invokes to get
Java side of the object wrapped as an iterable. This prevents returning multiple values - for example Synsets - in Ma
- So only tokens at a level are map-reduced and returned while the prevlevelsynsets (required for adding edges across
with a properietary asfer_pickle_load() and asfer_pickle_dump() functions that circumvents the python and java vagari
- With proprietary pickling and Map-Reduce functions, Recursive Gloss Overlap graph has been constructed in parallel.
- Proprietary pickling is done to a text file which is also added to repository. This file is truncated at the end of
- Subtlety here is that maximum number of tokens at a recursion level t = number_of_tokens_at_level_(t-1) * maximum_s
which grows as series = number_of_words*(1 + s + s^2 + s^3 + ... + s^t_max). Size of a document - number of words - c
- If size of the Spark cluster is O(f(d)*s^t_max), each recursion step is of time O(d*s^t_max/f(d)*s^t_max)=O(d/f(d))
- Maximum size of gloss per word (s) is also an upper-boundable constant. With constant d and t_max , size of cluster
- Example: For set of 1000 word documents with f(d)=log(d), max gloss size 5 and recursion depth 2, size of cluster i
- Above is just an estimate of approximate speedup achievable in Spark cluster. Ideally runtime in Spark cloud should
- Thus runtime upperbound in worst case is O(n*d*s^t_max/f(d)) for cluster size O(f(d)*s^t_max).
- If cluster autoscales based on number of documents also, size is a function of n and hence previous size bound is c

------------------------------------------------------------------------------------------------------------------------
202. (THEORY) Recursive Gloss Overlap, Cognitive and PsychoLinguistics and Language Comprehension
------------------------------------------------------------------------------------------------------------------------
Recursive Gloss Overlap algorithm constructs a graph from text documents. Presently WordNet is probably the only solu
get relations across words in a document. But the algorithm does not assume WordNet alone. Any future available algor

Intuition for Recursive Gloss Overlap for weighing natural language texts is from computational linguistics, Eye-Move

Psycholinguistics have the notion of event related potentials - when brain reacts excessively to anamolous words in n

A thought experiment of intrinsic merit versus prestige ranking:
Performance of an academic personality is measured first by accolades,awards,grades etc., which form the societal opi


References:
-----------
202.1 Circuits of the Mind - [Leslie Valiant] - http://dl.acm.org/citation.cfm?id=199266
202.2 Mind Grows Circuits - Lambda calculus and circuit modelling of mind - [Rina Panigrahy, Li Zhang] - http://arxiv
202.3 Psycholinguistics Electrified - EEG and SQUID Event Related Electric Potentials (ERP) peaking for anamolous wor
202.4 Word Associations and Evocations - http://hci.cse.ust.hk/projects/evocation/index.html
202.5 Combining WordNet, VerbNet, FrameNet - http://web.eecs.umich.edu/~mihalcea/papers/shi.cicling05.pdf
202.6 Computational Psycholinguistics - PoS Parsers - http://www.coli.uni-saarland.de/~crocker/courses/comp_psych/com
202.7 Brain Data for Psycholinguistics - http://personality.altervista.org/docs/14yg-al_brainsent@jlcl.pdf
202.8 ConceptNet 5 - http://conceptnet5.media.mit.edu/
202.9 Sanskrit WordNet - http://www.cfilt.iitb.ac.in/wordnet/webswn/
202.10 IndoWordNet - http://www.cfilt.iitb.ac.in/indowordnet/index.jsp
202.11 Brain Connectivity and Multiclass Hopfield Network - Associative memory - http://www.umiacs.umd.edu/~joseph/Wa
202.12 Text Readability Measures, Coherence, Cohesion - http://www.readability.biz/Coherence.html
202.13 MultiWordNet for European Languages - http://multiwordnet.fbk.eu/english/home.php
202.14 Coherence and Text readability indices (Coh-Metrix, FleschKincaid, Brain Overload etc.,) - http://lingured.in
202.15 Readability and WordNet - http://www.aclweb.org/anthology/O08-1
202.16. Semantic Networks (Frames, Slots and Facets) and WordNet - http://onlinelibrary.wiley.com/doi/10.1207/s155167
202.17 Text Network Analysis - Extracting a graph relation from natural language text - http://noduslabs.com/publicat
202.18 Text Network Analysis - Networks from Texts - http://vlado.fmf.uni-lj.si/pub/networks/doc/seminar/lisbon01.pdf
202.19 Six Degrees - Science in Connected Age - [Duncan J.Watts] - Pages 141-143 - Groundbreaking result by [Jon Klei
202.20 Align,Disambiguate,Walk distance measures - http://wwwusers.di.uniroma1.it/~navigli/pubs/ACL_2013_Pilehvar_Jur
202.21 Coh Metrix - Analysis of Text on Cohesion and Language - https://www.ncbi.nlm.nih.gov/pubmed/15354684 - quanti
202.22 Homophily in Networks - [Duncan J.watts] - Science in Connected Age (Pages 152-153) - https://arxiv.org/pdf/co

------------------------------------------------------------------------------------------------------------------------
203. Apache Spark MapReduce Parallel Computation of Interview Algorithm Recursive Gloss Overlap Graph Construction
- Commits as on 25 December 2015
------------------------------------------------------------------------------------------------------------------------
- Added more parallelism to python-src/InterviewAlgorithm/InterviewAlgorithmWithIntrinisicMerit_Crawl_Visual_Spark.py
for computing parents (backedges in the recursion) have been added - mapFunction_Parents(), reduceFunction_Parents(),
in python-src/InterviewAlgorithm/InterviewAlgorithmWithIntrinisicMerit_SparkMapReducer.py
- These functions take the previous level tokens as inputs instead if synsets

------------------------------------------------------------------------------------------------------------------------
204. Commits as on 28 December 2015
```

```
3379 ------------------------------------------------------------------------------------------
3380 - Updated MapReduce functions in Spark Recursive Gloss Overlap implementation
3381
3382 ------------------------------------------------------------------------------------------
3383 205. Commits as on 29 December 2015
3384 ------------------------------------------------------------------------------------------
3385 - Updated python-src/InterviewAlgorithm/InterviewAlgorithmWithIntrinisicMerit_Crawl_Visual_Spark.py for choosing para
3386 serial parents backedges computation based on a boolean function.
3387 - This is because Recursive tokens mapreduce computation (Spark_MapReduce) is faster than serial version.
3388 But serial parents computation is faster than parallel parents computation ironically(Spark_MapReduce_Parents).
3389 This happens on single node Spark cluster.  So, parents computation has been made configurable(serial or parallel)
3390 - Logs and Screenshots for various texts experimented have been added to testlogs/
3391 - python-src/InterviewAlgorithm/InterviewAlgorithmWithIntrinisicMerit_SparkMapReducer.py mapreduce uses best_matching
3392 disambiguation now which was commented earlier.
3393 - MapFunction_Parents() has a pickling problem in taking a tuple of synset objects as input arg due to which synsets
3394  causes the slowdown mentioned above compared to serial parents() version. MapFunction_Parents() has been rewritten t
3395 - Slowdown could be resolved on a huge cluster.
3396 - Thus this is a mix of serial+parallel implementation.
3397 - Logs,DOT file and Screenshots for mapreduced parents() computation
3398
3399 ------------------------------------------------------------------------------------------
3400 206. Commits as on 30 December 2015
3401 ------------------------------------------------------------------------------------------
3402 Some optimizations and redundant code elimination in python-src/InterviewAlgorithm/InterviewAlgorithmWithIntrinisicMe
3403
3404 ------------------------------------------------------------------------------------------
3405 207. Commits as on 31 December 2015
3406 ------------------------------------------------------------------------------------------
3407 - Spark Recursive Gloss Overlap Intrinsic Merit code has been optimized and some minutiae bugs have been resolved.
3408 - Map function for parents computation in Spark cluster has been updated to act upon only the previous recursion leve
3409 by proprietary pickling of the keyword into a file storage and loading it, and not as a function argument
3410 - Problems with previous level gloss tokens were almost similar to MapReduce functions of the recursion
3411 - New pickling file for parents computation in Spark has been added
3412 - logs and screenshots have been added to testlogs
3413 - With this, Spark Intrinsic Merit computation is highly parallellized apt for large clouds
3414 - Also a parents_tokens() function that uses gloss tokens instead of synsets has been added
3415 - New pickling dump() and load() functions have been added for keyword storage
3416 - pickling is synchronized with python threading lock acquire() and release(). Shouldn't be necessary because of
3417 Concurrent Read Exclusive Write (CREW) of keyword, but for safer side to prevent Spark-internal races.
3418
3419 ------------------------------------------------------------------------------------------
3420 208. Commits as on 1,2,3,4,5,6,7 January 2016
3421 ------------------------------------------------------------------------------------------
3422 - Added sections 53.14, 53.15 for HLMN and PARITY 3-SAT in 53
3423 - updated spidered text
3424 - best_matching_synset() enabled in mapreduce backedges computation which accurately disambiguates the graph. Spark s
3425 significantly slower than serial version of parents() computation with only python calls probably due to costly Pytho
3426 back-and-forth stream socket reads.
3427 - logs, DOT file and screenshots for single node Spark cluster have been added to testlogs/
3428
3429 ------------------------------------------------------------------------------------------
3430 209. Commits as on 8 January 2016
3431 ------------------------------------------------------------------------------------------
3432 - In InterviewAlgorithm/InterviewAlgorithmWithIntrinisicMerit_SparkMapReducer.py increased local threads to 2 (for du
3433 all SparkContexts instantiated
3434 - Added screenshot and logs for 2 local threads SparkContext mapreduce
3435 - Reference: Berkeley EdX Spark OpenCourse - https://courses.edx.org/c4x/BerkeleyX/CS100.1x/asset/Week2Lec4.pdf
3436
3437 ------------------------------------------------------------------------------------------
3438 210. Commits as on 10 January 2016
3439 ------------------------------------------------------------------------------------------
3440 NeuronRain Research (SourceForge) version 2016.1.10 released.
3441
3442 ------------------------------------------------------------------------------------------
3443 211. Commits as on 11 January 2016
3444 ------------------------------------------------------------------------------------------
3445 Added section 53.16 for an apparent contradiction between polysize and superpolynomial size among P/poly, Percolation
3446 special case of HLMN theorem.
3447
3448 ------------------------------------------------------------------------------------------------------------------------
3449 212. (FEATURE - DONE) Principal Component Analysis (PCA) and Singular Value Decomposition (SVD) - Commits as on 12 Ja
3450 ------------------------------------------------------------------------------------------------------------------------
3451 Python rpy2 wrapper implementation for :
3452         - Principal Component Analysis(PCA)
3453         - Singular Value Decomposition(SVD)
3454 which invoke R PCA and SVD functions and plot into 2 separate pdf files has been added to repository.
3455 Logs for an example have been added to testlogs/
3456
3457 ------------------------------------------------------------------------------------------------------------------------
3458 213. (FEATURE - DONE) Kullback-Leibler Divergence - Commits as on 17 January 2016
3459 ------------------------------------------------------------------------------------------------------------------------
```

```
Kullback-Leibler Divergence implementation:
-------------------------------------------
Approximate distance between 2 probability distributions with logs
in terms of weighted average distance represented as bits

-----------------------------------------------------------------------------------------------------------------------
214. (FEATURE - DONE) Basic Statistics - Norms and Median - Commits as on 19 January 2016
-----------------------------------------------------------------------------------------------------------------------
Implementation for basic statistics - L1, L2 norms, median etc.,

-----------------------------------------------------------------------------------------------------------------------
215. Commits as on 20 January 2016
-----------------------------------------------------------------------------------------------------------------------
- Updated AsFer Design Document for Psycholinguistics of Reading a Text document and Recursive Gloss Overlap
- Added Standard Deviation and Chi-Squared Test R functions to python-src/Norms_and_Basic_Statistics.py

-----------------------------------------------------------------------------------------------------------------------
216. (THEORY) Recursive Lambda Function Growth Algorithm - Psycholinguistic Functional Programming simulation of Huma
-----------------------------------------------------------------------------------------------------------------------
Example sentence:
California Gas Leak Exposes Growing Natural Gas Risks.

A left-right scan of the human reading groups the sentence into set of phrases and connectives and grows top-down glo
p1 - California Gas Leak
p2 - Exposes
p3 - Growing Natural Gas Risks

For each phrase left-right, meaning is created by recursive gloss overlap disambiguated graph construction top-down (
p1 - graph g1
p2 - graph g2
p3 - graph g3

Prefix graph construction and functional programming:
Above sentence has three prefix phrases and graphs- p1, p1-p2, p1-p2-p3 in order (and g1, g1-g2, g1-g2-g3). As reader

This formulation does not depend on just semantic graphs - graphs g1,g2 and g3 could be functional programming subrou
p1 - function g1  - california_gas_leak()
p2 - function g2  - exposes()
p3 - function g3  - growing_natural_gas_risks()
f3 = f2(f1(g1,g2),g3)

How functions f1,f2,g1,g2,g3 are internally implemented is subjective. These functions are dynamically created and ev

As an ideal human sense of "meaning" stored in brain is vague and requires a "Consciousness and Freewill" theory, onl

Learning theory perspective of the previous - From Linial-Mansour-Nisan theorem , class of boolean functions of n var

References:
-----------
216.1 Eye Movement Tracking - https://en.wikipedia.org/wiki/Psycholinguistics#Eye-movements
216.2 Eye Movements in Text Comprehension - Fixations, Saccades, Regressions - http://www.jove.com/video/50780/using-
216.3 Symbol Grounding Problem and learning from dictionary definitions - http://aclweb.org/anthology//W/W08/W08-2003
216.4 TextGraphs - graph representation of texts - https://sites.google.com/site/textgraphs2017/
216.5 Symbol Grounding Problem - http://users.ecs.soton.ac.uk/harnad/Papers/Harnad/harnad90.sgproblem.html - " ... (i
216.6 Understanding Natural Language - [Terry Winograd] - https://dspace.mit.edu/handle/1721.1/7095#files-area - Chap

-------------------------------------------------------------------------------
217. Commits as on 21 January 2016
-------------------------------------------------------------------------------
- Corrected Chi-squared test input args
- logs added to testlogs/

-------------------------------------------------------------------------------
218. Commits as on 27 January 2016
-------------------------------------------------------------------------------
Updated Sections 14 and 216.

-------------------------------------------------------------------------------
219. Commits as on 29 January 2016
-------------------------------------------------------------------------------
-  Uncommented both commandlines in cpp-src/asferpythonembedding.sh
-----------------------------------------------------------------------------------------------------------------------
220. (FEATURE - DONE)  Python-C++-VIRGOKernel and Python-C-VIRGOKernel boost::python and cpython implementations:
-----------------------------------------------------------------------------------------------------------------------
- It is a known idiom that Linux Kernel and C++ are not compatible.
- In this commit an important feature to invoke VIRGO Linux Kernel from userspace python libraries via two alternativ
- In one alternative, C++ boost::python extensions have been added to encapsulate access to VIRGO memory system calls
- In the other alternative, C Python extensions have been added that replicate boost::python extensions above in C -
works exceedingly well compared to boost::python.
- This functionality is required when there is a need to set kernel analytics configuration variables learnt by AsFer
dynamically without re-reading /etc/virgo_kernel_analytics.conf.
```

```
- This completes a major integration step of NeuronRain suite - request travel roundtrip to-and-fro top level machine
code and rock-bottom C linux kernel - bull tamed ;-).
- This kind of python access to device drivers is available for Graphics Drivers already on linux (GPIO - for accessi
- logs for both C++ and C paths have been added in cpp_boost_python_extensions/ and cpython_extensions.
- top level python scripts to access VIRGO kernel system calls have been added in both directories:
        CPython - python cpython_extensions/asferpythonextensions.py
        C++ Boost::Python - python cpp_boost_python_extensions/asferpythonextensions.py
- .so, .o files with build commandlines(asferpythonextensions.build.out) for "python setup.py build" have been added
in build lib and temp directories.
- main implementations for C++ and C are in cpp_boost_python_extensions/asferpythonextensions.cpp and cpython_extensi

- Schematic Diagram:
--------------------
        AsFer Python -----> Boost::Python C++ Extension ------> VIRGO memory system calls --------> VIRGO Linux Kerne
        /\                                                                                          V
         |                                                                                          |
         ----------------------------------------------<-----------------------------------------------
        AsFer Python -----> CPython Extensions ------> VIRGO memory system calls --------> VIRGO Linux Kernel Memory
         /\                                                                                         V
         |                                                                                          |
         ----------------------------------------------<-----------------------------------------------


-------------------------------------------------------------------------
221. Commits as on 2 February 2016
-------------------------------------------------------------------------
- Uncommented PyArg_ParseTuple() to read in the key-value passed from Python layer
- Unified key-value in Python layer and within CPython code with : delimiter
- added Py_BuildValue() to return vuid - VIRGO Unique ID - to python and commented virgo_free() so that a parallel co
can connect to kmem cache and do a virgo_get on this vuid - this is a precise scenario where Read-Copy-Update fits in
multiple versions can co-exist at a time
- Specific to NeuronRain Research version - experimental - for GRAFIT training materials:
        - separated VIRGO system calls into set and get kernel analytics python module functions
        - Python asferpythonextensions.py has been updated with 2 functions for get and set
        - Logs for this have been added along with a weird error noticed which is unrelated to this and
occurs in USB driver that too in Wi-Fi when USB device is not active
- All VIRGO system calls work in this segregated version similar to unified CPython
code in NeuronRain enterprise GitHub codebase

-------------------------------------------------------------------------------------------------------------------
222. (THEORY) Recursive Lambda Function Growth Algorithm - 216 elaborated with examples
-------------------------------------------------------------------------------------------------------------------
Boolean function learning and PAC learning are special cases of this Lambda function learning algorithm because any
intermediate lambda function can be a boolean function too.

Algorithms steps for English Natural Language Procssing - Parallel read:
-------------------------------------------------------------------------
- Approximate midpoint of the sentence is known apriori
- Language specific connectives are known apriori (e.g is, was, were, when, who etc.,)
- Sentence is recursively split into phrases and connectives and converted into lambda functions with balanced number
- Root of each subtree is unique name of the function

Example Sentence1:
-----------------
PH is infinite relative to random oracle.

Above sentence is translated into a lambda function composition tree as:
relative(is(PH, infinite), to(random,oracle))) which is a tree of depth 3

In this example every word and grammatical connective is a lambda function that takes some arguments and returns a pa

Example Sentence2:
-----------------
Farming and Farmers are critical to success of a country like India.

Above sentence is translated into a lambda function composition tree as:
Critical(are(and(Farming,Farmers)), a(success(to,of), like(country, India)))

This composition is slightly different from Part-of-Speech trees and inorder traversal of the tree yields original se
As functions are evaluated over time period, at any time point there is a partially evaluated composition prefix tree
rest of the sentence to be translated yet - these partial trees have placeholder templates that can be filledup by a
from rest of the sentence.

Each of these lambda functions can be any of the following but not limited to:
- Boolean functions
- Generic mathematical functions
- Neural networks (which include all Semantic graphs like WordNet etc.,)
- Belief propagated potentials assigned by a dictionary of English language and computed bottom-up
- Experimental theory of Cognitive Psycholinguistics - Visuals of corresponding words - this is the closest simulatic

        Mobile phones operate through towers in each cellular area that transmit signals.
```

```
with its per-word lambda function composition tree :

        in(operate(mobile phones, through(tower)), that(each(cellular area), transmit(signals)))

evokes from left-to-right visuals of Mobile phones, towers, a bounded area in quick succession based on user's person

The language of Sanskrit with Panini's Grammar fits the above lambda calculus framework well because in Sanskrit case
        Sambhala Grama Mukhyasya VishnuYashas Mahatmana: ... ( Chieftain of Sambhala Village, Vishnuyashas the great

can be rearranged and shuffled without altering the meaning, a precise requisite for lambda function composition tree

As an alternative to WordNet:
----------------------------
Each lambda function carries dictionary meaning of corresponding word with placeholders for arguments.
For example, for word "Critical" corresponding lambda function is defined as:
        Critical(x1,x2)
If dictionary meanings of critical are defined with placeholders:
        1) Disapproval of something -
        2)  - is Important to something -
there is a need for disambiguation and 2) has to be chosen based either on number of arguments or disambiguation usin
        x1 is important to something x2
Thus WordNet can be replaced by something more straightforward. These steps can be recursed top-down to expand the me

This algorithm accepts natural languages that are between Context Free Languages and Context Sensitive Languages.

References:
222.1 Charles Wikner - Introductory Sanskrit - http://sanskritdocuments.org/learning_tutorial_wikner/
222.2 Michael Coulson - Teach yourself - Sanskrit - http://www.amazon.com/Complete-Sanskrit-Teach-Yourself-Language/d


-------------------------------------------------------------------------------------------------------------------
223. (FEATURE - DONE) Crucial commits for Performance improvements and Cython build of Spark Interview algorithm impl
- Commits as on 4 February 2016
-------------------------------------------------------------------------------------------------------------------
- New setup.py has been added to do a Cythonized build of PySpark Interview algorithm MapReduce script
- Commandline for cython build: python setup.py build_ext --inplace
- This compiles python spark mapreduce script into a .c file that does CPython bindings and creates .o and .so files
- FreqDist has been commented which was slowing down the bootstrap
- MapReduce function has been updated for NULL object checks
- Thanks to Cython, Performance of Spark MapReduce has performance shootup by factor of almost 100x - Spark GUI job e
shown for this interview execution is ~10 seconds excluding GUI graph rendering which requires few minutes. This was
- With Cython ,essentially Spark-Python becomes as fast as C implementation.
- DOT file, Output log, Spark logs and Screenshot has been added
- This completes important benchmark and performance improvement for Recursive Gloss Overlap Graph construction on lo
cluster.

-----------------------------------------------------------------------------------------------
Commits as on 5 February 2016
-----------------------------------------------------------------------------------------------
-------------------------------------------------------------------------
224. Further optimizations to Spark-Cython Interview algorithm implementation:
-------------------------------------------------------------------------
- webspidered text for RGO graph construction updated
- PySpark-Cython build commandlines added as a text file log
- Commented unused python functions - shingling and jaccard coefficient
- threading.Lock() Spark thread pickling synchronizer variable made global in MapReducer
- renamed yesterday's logs and screenshot to correct datetime 4 February 2016
- Added a new special graph node ["None"] for special cases when no parents are found in backedges computation in Map
default root in the RGO graph as shown in matplotlib networkx graph plot
- Spark logs show time per RDD job - an example MapReduce job takes 335 milliseconds
- Tried Cythonizing InterviewAlgorithmWithIntrinisicMerit_Crawl_Visual_Spark.py but there is a serious cython compile
Matplotlib-PyQt. Hence those changes have been backed out.
- DOT file and Cython C file have been updated

-----------------------------------------------------------------------------------
225. Commits as on 8 February 2016
-----------------------------------------------------------------------------------
- PySpark-Cythonized Interview Algorithm implementation generated graph has been pruned to remove edges invoving "Non
added during Spark MapReduce based on a config variable.
- Logs and Screenshots for the above have been added to testlogs/
- Some benchmark results parsed from the Spark logs with commandline - "for i in `grep Finished InterviewAlgorithmWit
       - Total time in milliseconds - 99201
       - Number of jobs - 113
       - Average time per mapreduce job in milliseconds - 877
Thus a merit computation of sample document with few lines (33 keywords) requires ~100 seconds in local Spark cluster

-------------------------------------------------------------------------------------------------------------------
226. (THEORY) Graph Edit Distance in Interview Algorithm, PySpark-Cython Interview Algorithm benchmarks and Merit in
Lambda Function Growth Algorithm
-------------------------------------------------------------------------------------------------------------------
Interview Algorithm to assess merit, throughout this document, refers to 3 algorithms - Citation graph maxflow(which
```

```
3703
3704  Previous benchmarks for PySpark-Cython parallel interview algorithm implementation could scale significantly on a clu
3705
3706  Computing Intrinsic Merit of a text with Recursive Lambda Function growth could involve multitude of complexity notic
3707         - Depth of Composition tree for a text
3708         - Size of resultant lambda function composition e.g size of WordNet subgraphs, neural nets, complexity of cont
3709         - In essence, this reduces to a lowerbound of lambda function composition - boolean circuit lowerbounds are sp
3710
3711  ------------------------------------------------------------------------------------------------------------------
3712  227. (THEORY) Thought Versioning - ThoughtNet and EventNet - Psychophilosophical Digression - Might have
3713  some parallels in Philosophical Logic - related to points (70-79)
3714  ------------------------------------------------------------------------------------------------------------------
3715  (*) This postulates an experimental model for a "Logical Brain". It makes no assumptions about neural synapses, firin
3716
3717  (*) ThoughtNet is built over time from inception of life - books a person reads, events in life, interactions etc., -
3718
3719  (*) Thought Versioning System - tentatively named ThoughtNet - mentioned in 227 is a generalization of Evocation Word
3720
3721  (*) An approximate parallel to ThoughtNet versioning is the Overlay FileSystems in Unixen. Overlay filesystems store
3722
3723  (*) EventNet when unified with ThoughtNet makes a cosmic storage repository, all pervasive. Thoughts are stored with
3724
3725  (*) For example, an event with pleasant happening might be stored with high potential of positive polarity as against
3726
3727  (*) EventNet is a cosmic causality infinite graph with partaker nodes whereas ThoughtNet is per partaker node. This e
3728
3729  (*) EventNet is a parallelly created Causality graph - event vertices occur in parallel across universe and causation
3730
3731  (*) Previous evocation model with HMM or maximum potential need not be error-free - it can be derailed since human un
3732  ----------------------------------------------------------------
3733  Thought experiment for this defective disambiguation is as below
3734  ----------------------------------------------------------------
3735  (*) Example Sentence 1: "Love at first sight is the best". Considering two persons reading this sentence with drastic
3736
3737  (*) Example Sentence 2: "You are too good for any contest". This sentence has sarcastic tone for a human reader. But
3738
3739  (*) Example Poetry 3 - lambda evaluation with shuffled connectives, is shown below with nested parenthesization - Eac
3740  (("The time has come,") (the Walrus said),
3741  ("To talk of many things:)
3742  (Of shoes--and ships--and sealing-wax--)
3743  (Of cabbages--and kings--)
3744  And (why (the sea is boiling hot--))
3745  And (whether (pigs have wings.")))
3746  This lambda composition itself can be classified in classes viz., "time", "poetry", "sea" etc., (in special case of g
3747
3748  (*) Perfect design of Psycholinguistic text comprehension must involve ego-centric theory specific to each reader as
3749
3750  (*) Above exposition is required to get to the bottom of difficulties in formalising "what is meant by meaning" and "
3751
3752  (*) If ThoughtNet begot EventNet by anthropic principle, there is a possibility language was born before something ex
3753
3754  (*) Another counterexample: A cretan paradox like sentence - "This sentence is a lie" - which is true if it is false
3755
3756  (*) If there are two levels of Truth/Falsehoods similar to stratified realities defined in https://sites.google.com/s
3757          - This sentence is a lie - true - self-refuting and circular
3758          - This sentence is a lie - TRUE - not self-refuting because TRUE transcends "true"
3759          - This sentence is a lie - false - self-refuting and circular
3760          - This sentence is a lie - FALSE - not self-refuting because FALSE transcends "false"
3761
3762  (*) (QUITE PRESUMPTIVE, THIS MIGHT HAVE A PARALLEL IN LOGIC BUT COULDN'T FIND IT) In other words, the sentence "(It i
3763
3764  (*) Above philosophical dissection of meaning has striking similarities to quantum mechanics - If language has substr
3765
3766  (*) Levels of Truth are reminiscent of Tarski's Undefinability of Truth Theorem - https://plato.stanford.edu/entries/
3767
3768  ------------------------------------------------------------------------------------------------
3769  228. (THEORY) Bose-Einstein Condensate model for Recursive Gloss Overlap based Classification
3770  ------------------------------------------------------------------------------------------------
3771  Bose-Einstein condensate fitness model by [Ginestra Bianconi] described in 18.10 for any complex network graph is:
3772          Fitness of a vertex = 2^(-b*Energy) where b is a Bose-Einstein condensate function.
3773
3774  Here fitness of a vertex is defined as ability to attract links to itself - Star complexity. It is not known if abili
3775
3776  Applying the above model to a relational graph obtained from text document implies graph theoretic representation of
3777
3778  Analogy: Recursive Gloss Overlap graph can be construed as a 3 dimensional rubbersheet with low energy words as troug
3779
3780  Reference:
3781  ---------
3782  228.1 Linked [Albert Laszlo Barabasi] - Chapter 8 - Einstein's Legacy
3783
```

```
3784  ----------------------------------------------------------------------------------------------------
3785  229. (THEORY) Expander Graphs, Bose-Einstein Condensate Fitness model and Intrinsic Merit
3786  ----------------------------------------------------------------------------------------------------
3787  Edge expander graphs are defined with Cheeger's constant for minimum high edge boundary per vertex subset:
3788        minimum(|boundary(S)|/|S|), 1 < |S| < n/2
3789  where boundary(S) of a subset S of vertices of G (V(G)) is:
3790        boundary(S) = { (u,v) | u in S, v in V(G) \ S}
3791  From Bose-Einstein condensate fitness model, ability of a vertex to attract links increases with decreasing energy of
3792  boundary(S) expands with presence of low energy vertices. Thus Bose-Einstein condensate model translates into an edge
3793
3794  Measuring intrinsic merit (and therefore fitness if both are equivalent concepts) by edge expansion (Cheeger's consta
3795
3796  ----------------------------------------------------------------------------------------------------------------------
3797  230. (THEORY) Sentiment analysis and ThoughtNet Continuum - Fine grained emotions and evocation - related to 228
3798  ----------------------------------------------------------------------------------------------------------------------
3799  (*) ThoughtNet need not be a discrete hypergraph but a continuum of edges and nodes. Analogy: Set of countably infini
3800  (*) Each feature of a visual thought is a class stack node in ThoughtNet. Number of features per visual could be thou
3801  into a hyperedge across class stack nodes corresponding to feature vector vertices.
3802  ---------------------
3803  (*) Example schematic:
3804  ---------------------
3805                --------->Lambda computation---------
3806  returns       |                                  |  stores thought hyperedges
3807  evocatives    |                                  |
3808                ---------ThoughtNet <---------------
3809  (*) Colors of Thought hyperedges are determined by sentimentality of class stack nodes. The class stack nodes are col
3810        Edge sentiment color of ThoughtNet Hyperedge = weight_function(node colors in the hyperedge).
3811  (*) Node colors are priors. How these priors are decided for each class stack node is the mainstay of emotional infer
3812
3813  ----------------------------------------------------------------------------------------------------------------------
3814  231. VIRGO commits for AsFer Software Analytics with SATURN Program Analysis - 29 February 2016, 1 March 2016
3815  ----------------------------------------------------------------------------------------------------------------------
3816  ------------------------------------------------------------------------------
3817  Software Analytics - SATURN Program Analysis added to VIRGO Linux kernel drivers
3818  ------------------------------------------------------------------------------
3819  - SATURN (saturn.stanford.edu) Program Analysis and Verification software has been
3820  integrated into VIRGO Kernel as a Verification+SoftwareAnalytics subsystem
3821  - A sample driver that can invoke an exported function has been added in drivers - saturn_program_analysis
3822  - Detailed document for an example null pointer analysis usecase has been created in virgo-docs/VIRGO_SATURN_Program_
3823  - linux-kernel-extensions/drivers/virgo/saturn_program_analysis/saturn_program_analysis_trees/error.txt is the error
3824  - SATURN generated preproc and trees are in linux-kernel-extensions/drivers/virgo/saturn_program_analysis/preproc and
3825  linux-kernel-extensions/drivers/virgo/saturn_program_analysis/saturn_program_analysis_trees/
3826
3827  ----------------------------------------------------------------------------------------------------------------------
3828  232. (FEATURE-DONE) Commits as on 3 March 2016 - PERL WordNet::Similarity subroutine for pair of words
3829  ----------------------------------------------------------------------------------------------------------------------
3830  *) New perl-src folder with perl code for computing WordNet Distance with Ted Pedersen et al WordNet::Similarity CPAN
3831  *) Logs for some example word distances have been included in testlogs. Some weird behaviour for similar word pairs -
3832
3833  This leverages PERLs powerful CPAN support for text data processing. Also python can invoke PERL with PyPerl if neces
3834
3835  ----------------------------------------------------------------------------------------------------------------------
3836  233. (FEATURE-DONE) PERL WordNet::Similarity build and python issues notes
3837  ----------------------------------------------------------------------------------------------------------------------
3838  - NLTK WordNet corpora despite being 3.0 doesnot work with WNHOME setting (errors in exception files and locations of
3839  - Only Direct download from Princeton WordNet 3.0 works with WNHOME.
3840  - Tcl/Tk 8.6 is prerequisite and there is a build error "missing result field in Tk_Inter" in /usr/include/tcl8.6/tcl
3841        make Makefile.PL
3842        make
3843        make install
3844  in WordNet::Similarity with WNHOME set to /usr/local/WordNet-3.0.
3845  - PyPerl is no longer in active development - so only subprocess.call() is invoked to execute perl with shell=False.
3846
3847  ----------------------------------------------------------------------------------------------------------------------
3848  234. (THEORY) Dream Sentiment Analysis with ThoughtNet - related to 18 and 228
3849  ----------------------------------------------------------------------------------------------------------------------
3850  +Dream Analysis in http://cogprints.org/5030/1/NRC-48725.pdf (mentioned in 18.8) scores dreams based on polarity with
3851
3852  ----------------------------------------------------------------------------------------------------------------------
3853  235. (THEORY) Regularity Lemma and Ramsey Number of Recursive Gloss Overlap graph and ThoughtNet
3854  ----------------------------------------------------------------------------------------------------------------------
3855  Throughout this document, ranking of documents which is so far a pure statistical and machine learning problem is vie
3856        d(V1, V2) = |E(V1, V2)|/|V1||V2|
3857  where V1 and V2 are subsets of vertices V(G) for a graph G and E(V1,V2) are edges among subsets V1 and V2. Density is
3858
3859  ======================================================================================================================
3860  236. (FEATURE-DONE) Commits as on 11 March 2016 - Deep Learning Convolution - Multi Feature Convolution Map Kernel Fi
3861  ======================================================================================================================
3862  - Convolution Network supports Multiple Feature Maps (presently 3)
3863  - New example bitmap for feature recognition of pattern "3" inscribed as 1s has been introduced
3864  - Final neural network from Max pooling layer now is randomized with randint() based weights for
```

```
each of the 10 neurons
- Logs for this have been committed to testlogs
- Logs show a marked swing in Maxpooling map where the segments of pattern "3" are pronounced.
- Final neural layer shows a variegated decision from each neuron for corresponding 3 convolution maps

-----------------------------------------------------------------------------------------------------------
237. (FEATURE-DONE) Commits as on 14 March 2016
-----------------------------------------------------------------------------------------------------------
    - Some bugs resolved
    - Added one more example with no pattern
    - Convolution is computed for all 3 bitmap examples
    - Final neuron layer now is a function of each point in all maxpooling layers
    - The existence of pattern is identified by the final output of each of 10 neurons
    - Patterns 0 and 3 have a greater neural value than no pattern. Gradation of neural value indicates intensity of
    - Above is a very fundamental pattern recognition for 2 dimensional data. Sophisticated deconvolution is explaine
    - 3 logs for this commit have been included in testlogs/
    - random weighting has been removed.

-----------------------------------------------------------------------------------------------------------
238. (THEORY) Deep Learning Convolution Network and a boolean function learning algorithm of different kind
-----------------------------------------------------------------------------------------------------------
Deep Learning Convolution Network for Image Pattern Recognition elicits features from the image considered as 2-dimen

-----------------------------------------------------------------------------------------------------------
239. (FEATURE-DONE) Commits as on 15 March 2016
-----------------------------------------------------------------------------------------------------------
- DeepLearning BackPropagation implementation:
        - An example Software Analytics usecase for CPU and Memory usage has been included
        - Number of Backpropagation weight update iterations has been increased 10 fold to 3000000.
        - logs for some iterations have been included in testlogs/
- DeepLearning Convolution Network implementation:
        - Image patterns have been changed to 0, 8 and no-pattern.
        - Final neuron layer weights have been changed by squaring to scale down the output - this differentiates pat
        - logs for this have been included in testlogs/

-----------------------------------------------------------------------------------------------------------
240. (THEORY and IMPLEMENTATION) ThoughtNet and Reinforcement Learning
-----------------------------------------------------------------------------------------------------------
Reinforcement Learning is based on following schematic principle of interaction between environment and agent learnin

                              ---------> ThoughtNet context (environment) ------
        action (evocation)    |                                              |  reward and state transition (most
                              |<----------- Text Document (agent) ------<----- |

Schematic above illustrates text document study as reinforcement learning. During left-right reading of text, evocati
        - 1) First coin flip - Pr(doubling1) = 0.5, Pr(bankrupt1) = 0.5
        - 2) Second coin flip - Pr(bankrupt2) = Pr(bankrupt1)*0.5 = 0.25
        - 3) Third coin flip - Pr(bankrupt3) = Pr(bankrupt2)*0.5 = 0.125
        ....
By union bound probability of gambler going bankrupt after infinite number of coin flips is 1. This convergence to 1

A ThoughtNet Reinforcement Learning implementation has been committed as part of python-src/DeepLearning_Reinforcemen

References:
-----------
241.1 Reinforcement Learning - [Richard Sutton] - http://webdocs.cs.ualberta.ca/~sutton/book/the-book.html, https://w

-----------------------------------------------------------------------------------------------------------
241. (FEATURE-BENCHMARK-DONE) Interview Algorithm PySpark-Cython implementation benchmark with and without Spark conf
-----------------------------------------------------------------------------------------------------------
This benchmark is done with both:
InterviewAlgorithmWithIntrinisicMerit_Crawl_Visual_Spark.py
InterviewAlgorithmWithIntrinisicMerit_SparkMapReducer.py
both precompiled with Cython to create .c source files and .so libraries.

Number of keywords in newly crawled web page: 35

---------------------------
With spark-defaults.conf :
---------------------------
Execution 1:
------------
root@shrinivaasanka-Inspiron-1545:/media/shrinivaasanka/0fc4d8a2-1c74-42b8-8099-9ef78d8c8ea2/home/kashrinivaasan/Kris
> do
> sum=$(expr $sum + $i)
> done

root@shrinivaasanka-Inspiron-1545:/media/shrinivaasanka/0fc4d8a2-1c74-42b8-8099-9ef78d8c8ea2/home/kashrinivaasan/Kris
243058
root@shrinivaasanka-Inspiron-1545:/media/shrinivaasanka/0fc4d8a2-1c74-42b8-8099-9ef78d8c8ea2/home/kashrinivaasan/Kris
156
```

```
Per task time in milliseconds: ~1558

------------
Execution 2:
------------
root@shrinivaasanka-Inspiron-1545:/media/shrinivaasanka/0fc4d8a2-1c74-42b8-8099-9ef78d8c8ea2/home/kashrinivaasan/Kris
> do
> sum=$(expr $sum + $i)
> done
root@shrinivaasanka-Inspiron-1545:/media/shrinivaasanka/0fc4d8a2-1c74-42b8-8099-9ef78d8c8ea2/home/kashrinivaasan/Kris
520074
root@shrinivaasanka-Inspiron-1545:/media/shrinivaasanka/0fc4d8a2-1c74-42b8-8099-9ef78d8c8ea2/home/kashrinivaasan/Kris
312
Per task time in milliseconds: ~1666

-----------------------------
Without spark-defaults.conf
-----------------------------
Execution 3:
------------
root@shrinivaasanka-Inspiron-1545:/media/shrinivaasanka/0fc4d8a2-1c74-42b8-8099-9ef78d8c8ea2/home/kashrinivaasan/Kris
69691
root@shrinivaasanka-Inspiron-1545:/media/shrinivaasanka/0fc4d8a2-1c74-42b8-8099-9ef78d8c8ea2/home/kashrinivaasan/Kris
156
Per task time in milliseconds: ~446 which is twice faster than earlier benchmark done in February 2016 and almost 4 t

A spark-defaults.conf file has been committed to repository (Reference: http://spark.apache.org/docs/latest/configura

------------------------------------------------------------
242. Commits (1) as on 16,17 March 2016
------------------------------------------------------------
PySpark-Cython Interview Algorithm for Merit - Benchmark
------------------------------------------------------------
- 3 executions with and without spark-defaults.conf config settings have been benchmarked
- Cython setup.py has been updated to compile both InterviewAlgorithmWithIntrinisicMerit_Crawl_Visual_Spark.py and In
to create .c and .so files
- Benchmark yields some intriguing results:
        - Executions with spark-defaults.conf enabled are 4 times slower than executions without spark-defaults.conf
        - Execution without spark config is twice faster than earlier benchmark
- Logs and screenshots for above have been committed to testlogs
- Text document has been recrawled and updated (number of keywords almost same as previous benchmark)
- Spark config file has been committed

------------------------------------------------------------
PySpark-Cython Interview Algorithm for Merit - Benchmark
------------------------------------------------------------
- 3 executions with and without spark-defaults.conf config settings have been benchmarked
- Cython setup.py has been updated to compile both InterviewAlgorithmWithIntrinisicMerit_Crawl_Visual_Spark.py and In
to create .c and .so files
- Benchmark yields some intriguing results:
        - Executions with spark-defaults.conf enabled are 4 times slower than executions without spark-defaults.conf
        - Execution without spark config is twice faster than earlier benchmark
- Logs and screenshots for above have been committed to testlogs
- Text document has been recrawled and updated (number of keywords almost same as previous benchmark)
- Spark config file has been committed

------------------------------------------------------------
243. Commits (2) as on 16,17 March 2016
------------------------------------------------------------
More benchmarking of PySpark-Cython Intrinsic Merit computation
------------------------------------------------------------
- Enabled Spark RDD cacheing with cache() - storage level MEMORY
- Recompiled .c and .so with Cython
- uncommented both lines in Cython setup.py
- logs and screenshots for above have been committed in testlogs/
- locking.acquire() and locking.release() have been commented
- With this per task duration has been brought down to ~402 milliseconds on single node cluster. Ideally on a multinc
when tasks are perfectly distributable, this should be the runtime.

-------------------------------------------------------------------------------
With spark-defaults.conf - multiple contexts disabled and reuse worker enabled
-------------------------------------------------------------------------------
root@shrinivaasanka-Inspiron-1545:/media/shrinivaasanka/0fc4d8a2-1c74-42b8-8099-9ef78d8c8ea2/home/kashrinivaasan/Kris
246796
root@shrinivaasanka-Inspiron-1545:/media/shrinivaasanka/0fc4d8a2-1c74-42b8-8099-9ef78d8c8ea2/home/kashrinivaasan/Kris
156
Per task time milliseconds: ~1582

-----------------------------------------------------------------------------------------------
Without spark-defaults.conf - locking.acquire()/release() disabled and Spark RDD cacheing enabled
-----------------------------------------------------------------------------------------------
```

```
root@shrinivaasanka-Inspiron-1545:/media/shrinivaasanka/0fc4d8a2-1c74-42b8-8099-9ef78d8c8ea2/home/kashrinivaasan/Kris
248900
root@shrinivaasanka-Inspiron-1545:/media/shrinivaasanka/0fc4d8a2-1c74-42b8-8099-9ef78d8c8ea2/home/kashrinivaasan/Kris
618
Per task time milliseconds: ~402 (fastest observed thus far per task; Obviously something is wrong with spark-default

------------------------------------------------------------------------------------------------------------------------
244. (FEATURE-BENCHMARK-DONE) Interview Algorithm PySpark-Cython implementation benchmark with Spark configurables
- Commits as on 21 March 2016
------------------------------------------------------------------------------------------------------------------------
Following benchmark was done with a new spark-defaults.conf (committed as spark-defaults2.conf):

root@shrinivaasanka-Inspiron-1545:/media/shrinivaasanka/0fc4d8a2-1c74-42b8-8099-9ef78d8c8ea2/home/kashrinivaasan/Kris
965910
root@shrinivaasanka-Inspiron-1545:/media/shrinivaasanka/0fc4d8a2-1c74-42b8-8099-9ef78d8c8ea2/home/kashrinivaasan/Kris
618
Time per task in milliseconds: ~1562 (Again, with spark-defaults.conf with some additional settings, there is a block

------------------------------------------------------------------------------------------------------------------------
245. (THEORY) Learning a variant of Differential Privacy in Social networks
------------------------------------------------------------------------------------------------------------------------
Nicknames/UserIds in social media are assumed to be unique to an individual. Authentication and authorization with Pu
is assumed to guarantee unique identification. In an exceptional case, if more than one person shares a userid intent

------------------------------------------------------------------------------------------------------------------------
246. (FEATURE-DONE) Commits as on 24 March 2016 - Code Optimizations - Cacheing and Loop invariant - Interview Algori
------------------------------------------------------------------------------------------------------------------------
- New Cacheing datastructures have been included for storing already found previous level backedge vertices for each
already found gloss tokens. Presently these caches are python dictionaries only and in a distributed setting this can
object key-value stores like memcached.(InterviewAlgorithm/InterviewAlgorithmWithIntrinisicMerit_SparkMapReducer.py)
- There was an inefficient loop invariant computation for prevelevesynsets_tokens which has been moved to outermost w
- spark-default2.conf has been updated with few more Spark config settings
- Spidered text has been recrawled
- logs and screenshots for this commits have been included in testlogs/
- Cython .c , .so files have been rebuilt
- Cacheing above in someway accomplishes the isomorphic node removal by way of Cache lookup

------------------------------------------------------------------------------------------------------------------------
247. (FEATURE-BENCHMARK-DONE) Intrinsic Merit PySpark-Cython implementation - With and Without cacheing
- Commits as on 25 March 2016
------------------------------------------------------------------------------------------------------------------------
1.Cache md5 hashkey has been changed to be the complete tuple (tokensofprevlevel,keyword) - the rationale is to have
uniqueness to find backvertices for a keyword and gloss from previous recursion step.
2.Following benchmark was done by disabling and enabling lookupCache and lookupCacheParents boolean flags.
3.Cache values are "Novalue" string literals by default initialized in lambda and the Cache updates are not append()s
simple key-value assignments.
--
Number of keywords: 7
Without Cacheing: 9 minutes for all tasks (including NetworkX and graphics rendering times which is quite considerabl
With Cacheing: 9 minutes for all tasks (including NetworkX and graphics rendering times which is quite considerable)

Summation of individual task times in milliseconds and averaging sometimes gives misleadingly high duration per task
total execution is faster. Because of this start-end time duration is measured. For above example Cacheing has no eff
there are no repetitions cached to do lookup. Logs and screenshots with and without cacheing have been committed to t

------------------------------------------------------------------------------------------------------------------------
248. (THEORY) An alternative USTCONN LogSpace algorithm for constructing Recursive Gloss Overlap Graph
------------------------------------------------------------------------------------------------------------------------
Benchmarks above are selected best from multiple executions. Sometimes a mysterious slowdown was observed with both C
load which could be result of JVM Garbage Collection for Spark processes among others. These benchmarks are thus just

------------------------------------------------------------------------------------------------------------------------
249. Commits as on 29 March 2016
------------------------------------------------------------------------------------------------------------------------
- New subroutine for printing WordNet Synset Path between 2 words - this prints edges related by hypernyms and hypony
- an example log has been committed in testlogs/

From the example path in logs, distance is more of a metapath than a thesaurus-like path implemented in Recursive Glo

------------------------------------------------------------------------------------------------------------------------
250. (FEATURE-DONE) Commits (1) as on 30 March 2016 - MemCache Integration with Spark-Cython Intrinsic Merit Computat
------------------------------------------------------------------------------------------------------------------------
- In this commit, native python dictionary based cache lookup for Spark_MapReduce() and Spark_MapReduce_Parents() is
standard MemCache Distributed Object Cacheing get()/set() invocations.
- Logs and Screenshots for above have been committed to testlogs/
- Prerequisites are MemCached (ubuntu apt package) and Python-memcache (pip install) libraries
- a recrawled webpage has been merit-computed
- MemCached is better than native dict(s) because of standalone nature of Cache Listener which can be looked up from
anywhere on cloud

------------------------------------------------------------------------------------------------------------------------
```

```
251. (FEATURE-DONE) Commits (2) as on 30 March 2016 - Spark-Cython Intrinsic Merit computation with spark-defaults.cc
-------------------------------------------------------------------------------------------------------------------------
- spark-defaults2.conf has been enabled with extra java options for performance (reference: http://docs.oracle.com/ja
- lookupCache and lookupCacheParents boolean flags have been done away with because memcache has been enabled by defa
- With this spark logs show more concurrency and less latency. Dynamic allocation was in conflict with number of exec
- text document has been updated to have ~10 keywords. This took almost 7 minutes which looks better than one with sp
- Screenshot and logs have been committed in testlogs/

-------------------------------------------------------------------------------------------------------------------------
252. (FEATURE-DONE) Commits as on 1 April 2016
-------------------------------------------------------------------------------------------------------------------------
Loopless Map function for Spark_MapReduce()
-------------------------------------------------------------------------------------------------------------------------
- new Map function MapFunction2() has been defined which doesn't loop through all keywords per level, but does the gl
which is aggregated by Reduce step. MapFunction() does for loop for all keywords per level.
- a dictionary cacheing with graphcachelocal has been done for lookup of already gloss-tokenized keywords. This could
resets the graphcachelocal to a null which shouldn't have.
- This has a execution time of 6 minutes overall for all tasks compared to for-loop Map function in Spark_MapReduce()
- logs and screenshots for Loopless Map with and without graphcachelocal have been committed
- Rebuilt Cython .c, .o, .so, intrinsic merit log and DOT files

-------------------------------------------------------------------------------------------------------------------------
253. (THEORY) Hypercontractivity - Bonami-Beckner Theorem Inequality, Noise Operator and p-Norm of Boolean functions
-------------------------------------------------------------------------------------------------------------------------
Hypercontractivity implies that noise operator attenuates a boolean function and makes it close to constant function.
defined as:
        |Trhof|q <= |f|p where 0 <= rho <= sqrt((p-1)(q-1)) for some p-norm and q-norm
and in its expanded form as:
        sigma(rho^|S|*(1/2^n*sigma(|f(x)|^q*(-1)^(parity(xi))))^(1/q)*parity(S)) <= (1/2^n*sigma(|f(x)|^p)^(1/p)
Hypercontractivity upperbounds a noisy function's q-norm by same function's p-norm without noise. For 2-norm, by Pars

Reference:
----------
253.1 Bonami-Beckner Hypercontractivity - http://theoryofcomputing.org/articles/gs001/gs001.pdf

-------------------------------------------------------------------------------------------------------------------------
254. (THEORY) Hypercontractivity, KKL inequality,Social choice boolean functions  - Denoisification 3
-------------------------------------------------------------------------------------------------------------------------
For 2-norm, Bonami-Beckner inequality becomes:
        sigma[(rho^|S|*fourier_coeff(S)^2)^(2/2)] <= [1/2^n*(sigma(f(x)^p)^2/p)]

For boolean functions f:{0,1}^n -> {-1,0,1}, [Kahn-Kalai-Linial] inequality is special case of previous:
        For rho=p-1 and p=1+delta, sigma(delta^|S|*fourier_coeff(S)^2) <= (Pr[f!=0])^2/(1+delta)
where p-norm is nothing but a norm on probability that f is non-zero (because Pr[f!=0] is a mean of all possible f(x)

From 53.12 size of a noisy boolean function circuit has been bounded as: Summation(rho^|S|*fourier_coeff(S)^2) * 2^((
for |S| > t. From KKL inequality, 2-norm in LHS can be set to maximum of (Pr[f!=0])^2/(1+delta) in 53.12 and Size low
a denoisified circuit:
        (Pr[f!=0])^2/(1+delta) * 2^((t^(1/d)/20)-1) <= Size
which is exponential when delta=1 and Pr[f!=0] is quite close to 1 i.e f is 1 or -1 for most of the inputs.

If f is 100% noise stable, from Plancherel theorem:
       sigma(rho^|S|*f(S)^2) = 1, S in [n]
       For |S| < t=n, 1-Summation(rho^|S|*fourier_coeff(S)^2) <= 2*(Size)*2^(-t^(1/d)/20)
       But for rho=p-1 and p=1+delta, 1-Summation(delta^|S|*fourier_coeff(S)^2) >  1-(Pr[f!=0])^2/(1+delta) from KKL
       For |S| < t=n,  1-(Pr[f!=0])^2/(1+delta) < 1-Summation(rho^|S|*fourier_coeff(S)^2) <= 2*(Size)*2^(-t^(1/d)/20)
=>  0.5*(1-(Pr[f!=0])^2/(1+delta))*2^(t^(1/d)/20) < 0.5*(1-Summation(rho^|S|*fourier_coeff(S)^2))*2^(t^(1/d)/20) <= S

Again denoisification by letting rho and delta tend to 0 in above, places an exponential lowerbound, with constant up

PH in P^#P in P/Poly => P^#P=MA. (assumption: NC/Poly is in P/Poly because any logdepth, polysize bounded fanin circu
NC/log percolation circuit would imply collapse of polynomial hierarchy - RHS PH-complete is lowerbounded by LHS NC/l

Percolation boolean functions have a notion of revealment - that is, there exists a randomized algorithm correspondin
       Sigma(fourier_coeff(S)^2) <= revealment*constant*2-norm(f)

There is a special case in which size of advice list can be brought down to logn from n for Z(n) grid. Set of logn po

Reference:
----------
254.1 Randomized Algorithms and Percolation - http://math.univ-lyon1.fr/~garban/Slides/Newton.pdf

-------------------------------------------------------------------------------------------------------------------------
255. (THEORY) Majority Voting in Multipartisan Elections
-------------------------------------------------------------------------------------------------------------------------
Uptill now, P(Good) majority voting in this document describes only 2-candidate elections with a majority function vo
Noise stability of a voter's decision boolean function is assumed as a suitable measure of voter/voting error in RHS
generic RHS is when there is a multiway contest and error is bounded. P(Good) binomial summation still applies for mu
        - Individual Voter Judging Functions which are not boolean functions and
        - Generic Majority Function that needs to find most prominent index from an assorted mix of votes (Heavy hitt
        - There are three variables - number of variables per voter, number of voters and number of candidates (which
```

Voter judging functions have to choose an index within n candidates than simple 0-1. In a special case non-boolean fu

Reference:

255.1. Elections can be manipulated often - [Friedgut-Kalai-Nisan] - http://www.cs.huji.ac.il/~noam/apx-gs.pdf
255.2. Computational Hardness of Election Manipulation - https://www.illc.uva.nl/Research/Publications/Reports/MoL-26

---
256. Commits as on 4 April 2016

- Updated AsFer Design Document - KKL inequality and Denoisification
- Spark-Cython intrinsic merit computation has been md5hash enabled again since there were key errors with large stri
- logs and screenshots have been committed to testlogs/
- rebuilt Cython .c, .so files
- Spidered text has been changed - This took 28 minutes for ~40+ keywords better than 7 minutes for ~10 keywords earl

---
257. (FEATURE-BENCHMARK-DONE) Analysis of PySpark-Cython Intrinsic Merit Computation Benchmarks done so far
---
From 191 and 201 time complexity analysis of Recursive Gloss Overlap algorithm, for single node dualcore cluster foll
runtime bound:
       O(n*d*s^tmax/cpus)
where d is the number of keywords, n is the number of documents and s is the maximum gloss size per keyword. For last
MD5 hashkeys, Local Cacheing and Global Cacheing(MemCached) enabled number of words are 10 and 40. Corresponding runt
---
10 words
---
n=1
d=10
s=constant
tmax=2 (depth 2 recursion)
cpus=2 (dual core)
Runtime = 7 minutes = k*10*(s)^2/2
---
40 words
---
n=1
d=40
s=constant
tmax=2 (depth 2 recursion)
cpus=2 (dual core)
Runtime = 28 minutes = k*40*(s)^2/2
Ratio is 7/28 = k*10*(s)^2/2/k*40*(s)^2/2 = 0.25 a linear scaling in number of words which substantiates the theoreti

---
258. (THEORY) Major update to Computational Geometric PRAM-NC algorithm for Discrete Hyperbolic Factorization
---
PRAM-NC Algorithm for Discrete Hyperbolic Factorization in 34 above:

       34.1 LaTeX - http://sourceforge.net/projects/acadpdrafts/files/DiscreteHyperbolicPolylogarithmicSieveForIntege
       34.2 PDF - http://sourceforge.net/projects/acadpdrafts/files/DiscreteHyperbolicPolylogarithmicSieveForIntegerF

internally relies on [BerkmanSchieberVishkin] and other All Nearest Smaller Values PRAM algorithms for merging sorted

       259.1) There are 3 parallel computing models - Parallel Comparisons, Comparator Sorting Networks and PRAMs
       259.2) Section 2.4.1 on merging two increasing sequences of length n and m, n <= m in Page 19 gives an algorit
       259.3) Section 2.4.2 describes Batcher's and [AjtaiKomlosSzmeredi] Bitonic Sort algorithm which sorts n elemen
       259.4) Section 2.4.3 describes Cole's PRAM sorting algorithm which requires O(logn) steps and O(n) PRAM proces

Advantage of these older algorithms is they are implementable e.g Bitonic sort - https://en.wikipedia.org/wiki/Bitoni

References:

258.1 Bitonic Sorting - https://cseweb.ucsd.edu/classes/wi13/cse160-a/Lectures/Lec14.pdf

---
259. Commits as on 10 April 2016

Complement Function script updated with a new function to print Ihara Identity zero imaginary parts mentioned in
Complement Function extended draft:
       1. https://sourceforge.net/p/asfer/code/HEAD/tree/asfer-docs/AstroInferDesign.txt
       2. https://github.com/shrinivaasanka/asfer-github-code/blob/master/asfer-docs/AstroInferDesign.txt

Logs for this included in python-src/testlogs/

---
260. Commits (1) as on 12 April 2016

(FEATURE-DONE) Implementation for NC Discrete Hyperbolic Factorization with Bitonic Sort alternative (in lieu of unav
---

```
Bitonic Merge Sort Implementation of:
        - http://sourceforge.net/projects/acadpdrafts/files/DiscreteHyperbolicPolylogarithmicSieveForIntegerFactoriza
         - http://sourceforge.net/projects/acadpdrafts/files/DiscreteHyperbolicPolylogarithmicSieveForIntegerFactoriz

- Bitonic Sort is O((logn)^2) and theoretically in NC
- Bitonic Sort requires O(n^2logn) parallel comparators. Presently this parallelism is limited to parallel
exchange of variables in bitonic sequence for which python has builtin support (bitonic_compare()) - internally how i
multicore archs is not known.
- a shell script has been added which does the following (cpp-src/miscellaneous/DiscreteHyperbolicFactorizationUpperb
        - NC Factorization implementation with Bitonic Sort is C+++Python (C++ and Python)
        - C++ side creates an unsorted mergedtiles array for complete tesselated hyperbolic arc
(cpp-src/miscellaneous/DiscreteHyperbolicFactorizationUpperbound_Bitonic.cpp)
        - This mergedtiles is captured via output redirection and awk in a text file
cpp-src/miscellaneous/DiscreteHyperbolicFactorizationUpperbound_Bitonic.mergedtiles
        - Python bitonic sort implementation which is a modified version of algorithm in https://en.wikipedia.org/wik
requires the size of the array to be sorted in exponents of 2. Presently it is hardcoded as array of size 16384 - 2^1
(python-src/DiscreteHyperbolicFactorizationUpperbound_Bitonic.py). It outputs the sorted array. Vacant elements in ar

- Binary and logs for C++ and Python side have been committed
- Parallel comparators on large scale require huge cloud of machines per above bound which makes it NC.
-------------------------------------------------------------------------------------------------------
Commits (2) as on 12 April 2016
-------------------------------------------------------------------------------------------------------
- Compare step in Bitonic Sort for merged tiles has been Spark MapReduced with this separate script - ../python-src/D

References:
-----------
260.1 Parallel Bitonic Sort - http://www.cs.utexas.edu/users/plaxton/c/337/05s/slides/ParallelRecursion-1.pdf

-------------------------------------------------------------------------------------------------------
261. Commits as on 13 April 2016
-------------------------------------------------------------------------------------------------------
Spark Bitonic Sort - Imported BiDirectional Map - bidict - for reverse lookup (for a value, lookup a key in addition
lookup a value) which is required to avoid the looping through of all keys to match against a tile element.

-------------------------------------------------------------------------------------------------------
262. Commits as on 27 April 2016
-------------------------------------------------------------------------------------------------------
  Interim Commits for ongoing investigation of a race condition in Spark MapReduce for NC Factorization with Bitonic
-------------------------------------------------------------------------------------------------------------------
- The sequential version of bitonic sort has been updated to do away with usage of boolean flag up by 2 compare funct
- Spark NC implementation of Bitonic Sort for Factorization still has some strange behaviour in sorting. In progress
following in Compare-And-Exchange phase:
        - Comparator code has been rewritten to just do comparison of parallelized RDD set of tuples where each tuple
                (i, i+midpoint)
        - This comparison returns a set of boolean flags collect()ed in compare()
        - The variable exchange is done sequentially at each local node.
        - This is because Spark documentation advises against changing global state in worker nodes (though there are
        - There were some unusual JVM out of memory crashes, logs for which have been committed
        - some trivial null checks in complement.py
        - Bitonic Sort presents one of the most non-trivial cases for parallelism - parallel variable compare-and-exc
and present commits are a result of few weeks of tweaks and trial-errors.
        - Still the race condition is observed in merge which would require further commits.

-----------------------------------------------------------------------------
263. Commits as on 28 April 2016
-----------------------------------------------------------------------------
Interim commits 2 for ongoing investigation of race conditions in Spark Bitonic Sort:
-----------------------------------------------------------------------------
        - bidict usage has been removed because it requires bidirectional uniqueness, there are better alternatives
        - The variable exchange code has been rewritten in a less complicated way and just mimicks what sequential ve
in CompareAndExchange phase.
        - logs for Sequential version has been committed
        - number to factor has been set to an example low value
        - .coordinates and .mergedtiles files have been regenerated
        - Still the race condition remains despite being behaviourally similar to sequential version except the paral
        - Parallel comparator preserves immutability in local worker nodes

-------------------------------------------------------------------------------------------------------------
264. (THEORY) UGC(Unique Games Conjecture) and Special Case of Percolation-Majority Voting Circuits
-------------------------------------------------------------------------------------------------------------
264.1 Unique Games Conjecture by [SubhashKhot] is a conjecture about hardness of approximating certain constraint sat
264.2 Voter Boolean Functions can be written as :
        264.2.1 Integer Linear Programs : Voter Judging Boolean Functions are translated into an Integer Linear Progr
        264.2.2 and Constraint Satisfaction Problems : Voter expects significant percentage of constraints involving
264.3 Thus RHS of Majority Voting Circuit becomes a functional composition behemoth of indefinite number of individua
264.4 Approximating outcome of a majority voting (e.g. real life pre/post poll surveys) depends thus on following fac
        264.4.1 Correctness - Ratio preserving sample - How reflective the sample is to real voting pattern e.g a 60%
        264.4.2 Hardness - Approximation of Constraint Satisfaction Problems and UGC.

References:
```

```
4351 ----------
4352 264.5 Unique Games Conjecture and Label Cover - https://en.wikipedia.org/wiki/Unique_games_conjecture
4353 264.6 Design of Approximation Algorithms - Page 442 - http://www.designofapproxalgs.com/book.pdf
4354
4355
4356 ------------------------------------------------------------------------
4357 265. Commits 1 as on 29 April 2016
4358 ------------------------------------------------------------------------
4359 Interim commits 3 for Bitonic Sort Spark NC Factorization Implementation
4360 ------------------------------------------------------------------------
4361 - AsFer Design Document updated for already implemented NVIDIA CUDA Parallel C reference code for bitonic sort on hyp
4362 - Bitonic Sort C++ code updated for max size of sort array - reduced from 16384 to 256
4363 - Bitonic Sort Python Spark implementation updated for returning a tuple containing index info also instead of plain
4364 because Spark collect() after a comparator map() does not preserve order and index information was lost.
4365 - if clauses in exchange phase have been appropriately changed
4366 - Still there are random mysterious jumbled tuples which will be looked into in future commits
4367 - This implementation becomes redundant because of already existing better NVIDIA CUDA Parallel C implementation and
4368 is of academic interest only.
4369
4370 ------------------------------------------------------------------------
4371 Final commits for Bitonic Sort Spark NC Factorization Implementation
4372 ------------------------------------------------------------------------
4373 - Indeed Bitonic Sort from previous commit itself works without any issues.
4374 - Nothing new in this except reduced sort array of size just 64 for circumventing frequent Py4Java Out of Memory erro
4375 - Logs for this have been committed in testlogs/
4376 - the debug line "merged sorted halves" is the final sorted tiles array
4377 - .coordinates and .mergedtiles files have been updated
4378
4379 ------------------------------------------------------------------------
4380 266. Commits as on 30 April 2016
4381 ------------------------------------------------------------------------
4382 - Printed return value of bitonic_sort() in sorted. Earlier it was wrongly printing globalmergedtiles a global state
4383 anymore
4384 - logs for bitonic sort of tesselated hyperbolic arc
4385
4386 ------------------------------------------------------------------------
4387 267. Commits as on 1 May 2016
4388 ------------------------------------------------------------------------
4389 - The Map step now does both Compare and Exchange within local worker nodes for each closure and result is returned i
4390 - The driver collect()s and just assigns the variables based on comparator boolean field in tuple which are already e
4391 local worker
4392
4393 ---------------------------------------------------------------------------
4394 268. Commits as on 2 May 2016
4395 ---------------------------------------------------------------------------
4396 Accumulator support for Bitonic Sort mapreduce globalmergedtiles
4397 ---------------------------------------------------------------------------
4398 - Spark context has been parametrized and is passed around in the recursion so that single spark context keeps track
4399 accumulator variables
4400 - globalmergedtiles_accum is the new accumulator variable which is created in driver and used for storing distributed
4401 - AccumulatorParam has been overridden for Vectors as globals
4402 - Spark does not permit however, at present, mutable global state within worker tasks and only driver creates the acc
4403 - muting or accessing accumulator.value within worker raises Exception: Can't access value of accumulator in worker t
4404 - Logs for above have been committed to testlogs/
4405 - Because of above limitation of Spark in global mutability, accumulator globalmergedtiles is updated only outside th
4406 - Support for global mutables, if Spark has it, would make compare-and-exchange phase closer in behaviour to an NVIDI
4407 - Documentation on this necessary feature is scarce and some suggest use of datastores like Tachyon/Alluxio for distr
4408 which is non-trivial. This cloud parallel sort implementation achieves thus almost ~90% of NVIDIA CUDA parallel biton
4409 CompareAndExchange is done in parallel but assigning CompareAndExchange results is done sequentially.
4410
4411 ---------------------------------------------------------------------------
4412 269. Commits 1 as on 3 May 2016
4413 ---------------------------------------------------------------------------
4414 New threading function assign_compareexchange_multithreaded() has been implemented which is
4415 invoked in lieu of sequential for loop assignment , by creating a thread for each pair of (i,i+midpoint) compare-exch
4416 of accumulator globalmergedtiles.This thread function assigns the Spark Mapreduce result of Compare and Exchange.
4417
4418 Mulithreading is an alternative for global state mutability in the absence of Spark support
4419 and it does not require any third party in-memory cacheing products.
4420
4421 Coordinates are also shuffled first few elements of which correspond to number to factor N - these are
4422 the factors found finally. Multithreaded assignment maps to a multicore parallelism. Logs for this have been
4423 committed to testlogs with and without Coordinates Shuffling.
4424
4425 ---------------------------------------------------------------------------
4426 270. Commits 2 as on 3 May 2016
4427 ---------------------------------------------------------------------------
4428 Number to factorize has been changed to 147. The mergedtiles array has been set to size 256. The logs for this have b
4429
4430 sorted= [147, 147, 147, 147, 147, 147, 147, 147, 147, 147, 147, 147, 147, 146, 146, 146, 146, 146, 146, 146, 145, 145
4431 globalcoordinates= [1, 21, 21, 49, 49, 7, 147, 147, 3, 3, 3, 3, 7, 73, 73, 147, 2, 2, 145, 2, 144, 5, 5, 29, 29, 5, 2
```

```
sorted globalmergedtiles accumulator version:  =  [147, 147, 147, 147, 147, 147, 147, 147, 147, 147, 147, 147, 147, 1
16/05/03 18:33:06 INFO SparkUI: Stopped Spark web UI at http://192.168.1.101:4040
16/05/03 18:33:06 INFO DAGScheduler: Stopping DAGScheduler
16/05/03 18:33:06 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!

----
Factors of 147 are [1,21,49,7,147,3] from globalcoordinates and globalmergedtiles(corresponding to 147 elements) prin

-----------------------------------------------------------------------------------------
271. (THEORY) Integer Partitions, Riemann Sums, Multipartisan Majority Voting - a special case
-----------------------------------------------------------------------------------------
From https://sites.google.com/site/kuja27/IntegerPartitionAndHashFunctions_2014.pdf, Integer Partitions when visualiz
in discrete sense, correspond to Riemann Sums Discrete Area Integral i.e Each part in the partition slices the area u
by topmost points of each part in partition. This curve can be perturbed to arbitrary shape without altering the area
words each perturbed curve corresponds to a partition of N where N is the area integral value for the curve. A striki

-----------------------------------------------------------------------------------------
272. Commits as on 4 May 2016
-----------------------------------------------------------------------------------------
Sequence Mining of Prime Numbers as binary strings
-----------------------------------------------------------------------------------------
- First 10000 prime numbers have been written to a text file in binary notation in complement.py
- SequenceMining.py mines for most prominent sequences within prime binary strings - a measure of patterns in prime d
- Logs for this have been committed to testlogs/

-----------------------------------------------------------------------------------------
273. Commits as on 11 May 2016
-----------------------------------------------------------------------------------------
Sequence Mining in Prime binary strings has been made sophisticated:
        - Prints each binary string sequence pattern in decimals upto maximum of 15-bit sequences mined from first 10
        - This decimals are written to a file PatternInFirst10000Primes.txt
        - Approx.py R+rpy2 has been invoked to print an R function plotter of this decimal pattern in prime binary st
        - The function plots the sequences in Y-axis and length of sequences in ascending order in X-axis(with decrea
        - As can be seen from pdf plot, the sequences show dips amongst peaks periodically.

-----------------------------------------------------------------------------------------
274. Commits as on 17 May 2016
-----------------------------------------------------------------------------------------
Mined Class Association Rule Search Script for Weather Forecast has been updated to search for possible heavy precipi
for date range 17 May 2016 to 30 May 2016. Logs for this have been committed which show a prominent celestial rule ma

-----------------------------------------------------------------------------------------------------------------------
275. (THEORY) Non-boolean Social Choice Functions, Sensitivity, Stability and Multipartisan Majority Voting - 256 con
-----------------------------------------------------------------------------------------------------------------------
Let f be a voter decision function which takes in x1,x2,...xm as decision variables and outputs a vote for candidate
generalizes voter boolean judging functions for 2 candidates to arbitrarily large number of contestants. Assuming f t
function and parameters to be continuous (as opposed to discrete binary string SAT assignments to variables in boolea
generalizes Satisfiability problem to continuous, multivalued and non-boolean functions. This allows the candidate in
continuous too. The function plot of this (candidate index versus decision string) would resemble any continuous math
Stability which is: Pr[f(s) == f(t)] - Pr[f(s) != f(t)] for randomly correlated s=(x11,x12,...,x1m) and t=(x21,x22,x2
correlation (measured by usual euclidean distance between s and t) could be uncomputable value in terms of mean but c
as difference in ratio of length integral of function arc segments with high curvature and low curvature (where funct
and relatively flat).  Sensitivity follows accordingly. This requires first derivative of the function to find out lo
From Gibbard-Satterthwaite theorem any social choice function for more than 3 candidates has a non-zero probability o
negligible in practice. Probably this implies atleast indirectly that the stability of any non-boolean social choice
candidates > 3 hinting at divergence of Pr(Good) Binomial series. This function plot should not be confused with func
272 for votes partitioned across candidates which reduces to Riemann Sum. Unlike boolean sensitivity/stability, non-b
satisfiability because an appropriate assignment to (x1,x2,x3,.....,xm) has to be found so that f(x1,x2,x3,.....,xm)=
This is precisely curve-line intersection problem - curve f(x1,x2,x3,...,xn) and line y=c - in computational geometry
(a1,a2,a3,...,az) are the satisfying assignments which choose candidate c. Contrasting this with boolean kSAT which i
non-boolean kSAT has polynomial time computational geometric algorithms to find satisfying intersection points. It is
a sharp threshold phenomenon in computational hardness of satisfiability - an easy polytime non-boolean kSAT becomes
Assuming that real-life voters have non-boolean decision functions only, the RHS circuit of Pr(Good) majority voting
satisfiability realm alone. Non-boolean voting decision functions can be specialized for 2 candidates special case to
values to decision variables. This is plausible because Linear Programming (non-boolean) is polytime while 0-1 Intege
(boolean) is NP-complete.  Because of Non-boolean Social Choice functions (e.g Linear Programs with Constraints which

References:
----------
275.1 Intersection detection - https://www.cs.umd.edu/~mount/Papers/crc04-intersect.pdf
275.2 Transversal Intersection of two curves - Newton iteration and Cramer's rule - https://en.wikipedia.org/wiki/Int

-----------------------------------------------------------------------------------------------------------------------
276. (FEATURE-DONE) Commits as on 30 May 2016 - Continued from 220
-----------------------------------------------------------------------------------------------------------------------
VIRGO CloudFS system calls have been added (invoked by unique number from syscall_32.tbl) for C++ Boost::Python inter
VIRGO Linux System Calls. Switch clause with a boolean flag has been introduced to select either VIRGO memory or file
kern.log and CloudFS textfile Logs for VIRGO memory and filesystem invocations from AsFer python have been committed

-----------------------------------------------------------------------------------------------------------------------
277. (FEATURE-DONE) Commits as on 31 May 2016 - Continued from 220 and 276
```

```
4513 ----------------------------------------------------------------------------------------------------------------------
4514 Python CAPI interface to NEURONRAIN VIRGO Linux System Calls has been updated to include File System open, read, writ
4515 Rebuilt extension binaries, kern.logs and example appended text file have been committed to testlogs/. This is exactl
4516 commits done for Boost::Python C++ interface. Switch clause has been added to select memory or filesystem VIRGO sysca
4517
4518 ----------------------------------------------------------------------------------------------------------------------
4519 278. (FEATURE-DONE) Commits as on 7 June 2016
4520 ----------------------------------------------------------------------------------------------------------------------
4521 - getopts implementation for commandline args parsing has been introduced in Maitreya textclient rule search python s
4522 - an example logs for possible high precipitation longitude/latitudes in future dates - July 2016 - predicted by sequ
4523 rules from past data has been added to testlogs/
4524
4525 root@shrinivaasanka-Inspiron-1545:/home/shrinivaasanka/Maitreya7_GitHub/martin-pe/maitreya7/releases/download/v7.1.1/
4526 {  --date="2016-7-2 10:10:10 5"  --location=" x 77:0:0 7:0:0 " --planet-list } - There is a Class Association Rule ma
4527 {  --date="2016-7-3 10:10:10 5"  --location=" x 77:0:0 7:0:0 " --planet-list } - There is a Class Association Rule ma
4528 {  --date="2016-7-4 10:10:10 5"  --location=" x 77:0:0 7:0:0 " --planet-list } - There is a Class Association Rule ma
4529 {  --date="2016-7-5 10:10:10 5"  --location=" x 77:0:0 7:0:0 " --planet-list } - There is a Class Association Rule ma
4530 {  --date="2016-7-6 10:10:10 5"  --location=" x 77:0:0 7:0:0 " --planet-list } - There is a Class Association Rule ma
4531 {  --date="2016-7-11 10:10:10 5"  --location=" x 77:0:0 7:0:0 " --planet-list } - There is a Class Association Rule m
4532 {  --date="2016-7-13 10:10:10 5"  --location=" x 77:0:0 7:4:0 " --planet-list } - There is a Class Association Rule m
4533 {  --date="2016-7-14 10:10:10 5"  --location=" x 77:0:0 7:4:0 " --planet-list } - There is a Class Association Rule m
4534
4535 ----------------------------------------------------------------------------------------------------------------------
4536 279. (FEATURE-DONE) Commits - 23 June 2016 - Recurrent Neural Network Long Term Short Term Memory - Deep Learning - I
4537 ----------------------------------------------------------------------------------------------------------------------
4538 A very minimal python implementation of LSTM RNN based on Schmidhuber-Hochreiter LSTM has been added to
4539 already existing AsFer algorithms repertoire. LSTM RNN has gained traction in recent years in its variety of
4540 applications in NLP, Speech Recognition, Image Pattern Recognition etc., The logs for this implementation show
4541 a convergence of final output layer in 25th iteration itself (out of 10000). Learning the weights requires
4542 other algorithms like Gradient Descent, Backpropagation (and there are already known limitations in weight learning w
4543
4544 ----------------------------------------------------------------------------------------------------------------------
4545 280. (FEATURE-DONE) Commits - 23 June 2016 - Minimal Reinforcement Learning (Monte Carlo Action Policy Search) implem
4546 ----------------------------------------------------------------------------------------------------------------------
4547 Reinforcement Learning has been implemented which changes state based on environmental observation and an appropriate
4548 chosen for observation by Monte Carlo Random Policy Search based on which rewards for each transitions are accumulate
4549 Logs for 3 consecutive executions of Reinforcement Learning have been committed with differing total rewards gained.
4550 Input observations are read from text file ReinforcementLearning.input.txt.
4551
4552 Reference: Richard Sutton - http://webdocs.cs.ualberta.ca/~sutton/papers/Sutton-PhD-thesis.pdf
4553
4554 ----------------------------------------------------------------------------------------------------------------------
4555 281. (FEATURE-DONE) Commits - 24 June 2016 - ThoughtNet Reinforcement Learning implementation
4556 ----------------------------------------------------------------------------------------------------------------------
4557 ThoughtNet based Reinforcement Learning Evocation has been experimentally added as a clause in python-src/DeepLearnin
4558
4559 ----------------------------------------------------------------------------------------------------------------------
4560 282. (FEATURE-DONE) Commits - 26 June 2016 - ThoughtNet File System Storage
4561 ----------------------------------------------------------------------------------------------------------------------
4562 ThoughtNet text files have been stored into a filesystem backend. It is read and eval()-ed into list of edges and hyp
4563 Separate ThoughtNet directory has been created for these text files.
4564
4565 ------------------------------------------------------------------------------------------------
4566 283. ThoughtNet growth and Evocation Reinforcement Learning algorithm (not feasible to implement exactly):
4567 ------------------------------------------------------------------------------------------------
4568 loop_forever
4569 {
4570 When an observation stimulus from environment arrives:
4571        (*) Sensory instruments that simulate eye, ear, nose, etc., receive stimuli from environment (thoughts, music,
4572 and convert to sentential form - this is of the order of billions - and appended to list in ThoughtNet_Edges.txt
4573        (*) Sentences are classified into categories (for example, by maximum core numbers in definition wordnet subgr
4574        (*) observation is split to atomic units - tokenized - or classified and each unit is lookedup in ThoughtNet_H
4575 }
4576
4577 ----------------------------------------------------------------------------------------------------------
4578 284. (FEATURE-DONE) Auto Regression Moving Average - ARMA - Time Series Analysis for Stock Ticker Stream
4579 ----------------------------------------------------------------------------------------------------------
4580 Commits - 27 June 2016
4581 ----------------------------------------------------------------------------------------------------------
4582 1. Time Series Analysis with AutoRegressionMovingAverage of Stock Quote streamed data has been implemented (R functic
4583 show the actual data and projected quotes by ARMA for few iterations have been committed to testlogs. This ARMA proje
4584 2. ARMA code implements a very basic regression+moving averages. Equation used is same though not in usual ARMA forma
4585 (1-sigma())X(t) = (1+sigma())
4586 3. Also committed is the R plot for SequenceMined Pattern in first 10000 prime numbers in binary format (DJIA approx
4587
4588 ------------------------------------------------------------------------------------------
4589 285. (FEATURE-DONE) Commits 1 - 28 June 2016 - Neo4j ThoughtNet Hypergraph Database Creation
4590 ------------------------------------------------------------------------------------------
4591 1.New file ThoughtNet_Neo4j.py has been added to repository which reads the ThoughtNet edges and hypergraph text file
4592 Neo4j Graph Database through py2neo client for Neo4j.
4593 2.Neo4j being a NoSQL graph database assists in querying the thoughtnet and scalable.
```

```
3.ThoughtNet text files have been updated with few more edges and logs for how Neo4j graph database for ThoughtNet lo
committed to testlogs/

-----------------------------------------------------------------------------------------------
286. (FEATURE-DONE) Commits 2 - 28 June 2016 -  ThoughtNet Neo4j Transactional graph creation
-----------------------------------------------------------------------------------------------
1.transactional begin() and commit() for graph node and edges creation has been
included.
2.This requires disabling bolt and neokit disable-auth due to an auth failure config issue.
3.Logs and a screenshot for the ThoughtNet Hypergraph created in GUI (http://localhost:7474) have been committed to t

-------------------------------------------------------------------------------------------------------------------
287. (FEATURE-DONE) Commits - 29 June 2016 - ThoughtNet and Reinforcement Deep Learning
-------------------------------------------------------------------------------------------------------------------
Major commits for ThoughtNet Hypergraph Construction and Reinforcement Learning from it
-----------------------------------------------------------------------------------------
1. python-src/DeepLearning_ReinforcementLearningMonteCarlo.py reads from automatically generated ThoughtNet Hypergrap
by python-src/ThoughtNet/Create_ThoughtNet_Hypergraph.py in python-src/ThoughtNet/ThoughtNet_Hypergraph_Generated.txt
2. Separate Recursive Gloss Overlap CoreNumber and PageRank based Unsupervised Classifier has been implemented in pyt
3. python-src/ThoughtNet/ThoughtNet_Neo4j.py reads from automatically generated ThoughtNet in python-src/ThoughtNet/T
4. Logs for above have been committed to respective testlogs/ directories
5. Compared to human evaluated ThoughtNet Hypergraphs in python-src/ThoughtNet/ThoughtNet_Hypergraph.txt, generated p
6. python-src/ReinforcementLearning.input.txt and python-src/ThoughtNet/ThoughtNet_Edges.txt have been updated manual

-------------------------------------------------------------------------------------------------------------------
288. (FEATURE-DONE) Schematic Diagram for ThoughtNet Reinforcement Evocation - approximate implementation of 283
-------------------------------------------------------------------------------------------------------------------

        Environment -------(stimuli)------> Sensors -----> Create_ThoughtNet_Hypergraph.py ----> RecursiveGlossOverla
                                                                                              |
                                                                                              V
                                                                                              |
                                                                                              V
                        Observations----------------> DeepLearning_ReinforcementLearningMonteCarlo.py ---------> Evoc

-----------------------------------------------------------------------------------------------
289. (FEATURE-DONE) Commits - 30 June 2016 - Sentiment Scoring of ThoughtNet edges and Sorting per Class
-----------------------------------------------------------------------------------------------
1. Create_ThoughtNet_Hypergraph.py has been changed to do sentiment scoring which is a nett of positivity, negativity
thought hyperedge and to sort the per-class key list of hyperedges descending based on sentiment scores computed.
2. python-src/ReinforcementLearning.input.txt which is the input observation stimulus and python-src/ThoughtNet/Thoug
3. Hypergraph created for this is committed at python-src/ThoughtNet/ThoughtNet_Hypergraph_Generated.txt with logs in

This completes ThoughtNet Classifications based Evocative Inference implementation minimally.

-----------------------------------------------------------------------------------------------
290. (FEATURE-DONE) Commits - 1 July 2016 - Reinforcement Learning Bifurcation
-----------------------------------------------------------------------------------------------
Reinforcement Learning code has been bifurcated into two files than having in if-else clause, for facilitating future
one for MonteCarlo and the other for ThoughtNet Evocation.  Logs for these two have been committed to testlogs.

-----------------------------------------------------------------------------------------------
291. (THEORY) Logical Time and ThoughNet (related to EventNet 70-79)
-----------------------------------------------------------------------------------------------
ThoughtNet Hypergraph is multiplanar and when seen from elevation it resembles billions of skyscraper category towers
HyperEdges which are classified onto these categories. This gives an alternative route to define Time. Each tower rep


As a working example, following edges were evoked when word "economic" was uttered:
================================================================================
Observation: economic
evocative thought (reward) returned(in descending order of evocation potential): The HDI was created to emphasize tha
evocative thought (reward) returned(in descending order of evocation potential): We need an SPI and we need to unders
evocative thought (reward) returned(in descending order of evocation potential): Social progress depends on the polic
================================================================================
Only the top most evocative edge (sorted based on sentiment scores) has the word "economics" in it while the other tw

A practical ThoughtNet storage could be of billions of edges based on experiential learning of an individual over a p
        - Word: Sensory receptors perceive stimuli - events
        - Mind: stimuli evoke thoughts in past
        - Action: Evocative thought is processed by intellect and inspires action - action is a lambda evaluation of
################################################################################

        Senses(Word) <---------> Mind(Evocation) <---------> Action(Intellect)
         ^                                                   ^
         |<--------------------------------------------------->|

and above is an infinite cycle. Previous schematic maps interestingly to Reinforcement Learning(Agent-Environment-Act
################################################################################
In this aspect, ThoughtNet is a qualitative experimental inference model compared to quantitative Neural Networks.

```

It is assumed in present implementation that every thought edge is a state implicitly, and the action
for the state is the "meaning" inferred by recursive lambda evaluation (lambda composition tree evaluation for a natu
language sentence is not implemented separately because its reverse is already done through closure of a RGO graph in
code in NeuronRain AsFer. Approximately every edge in Recursive Gloss Overlap wordnet subgraph is a lambda function w
two vertices as operands which gives a more generic lambda graph composition of a text)

---

292. (THEORY) Recursive Lambda Function Growth Algorithm and Recursive Gloss Overlap Graph - 216 Continued

---

Lambda Function Recursive Growth algorithm for inferring meaning of natual language text approximately, mentioned in
on creation of lambda function composition tree top-down by parsing the connectives and keywords of the text. Alterna
parsing is to construct the lambda function composition graph instead from the Recursive Gloss Overlap WordNet subgra
each edge is a lambda function with two vertex endpoints as operands (mentioned in 291). Depth First Traversal of thi
evaluates a lambda function f1 of an edge, f1 is applied to next edge vertices in traversal to return f2(f1), and so
fn(fn-1(fn-2(.....(f2(f1))...) which completes the recursive composition. Which one is better - tree or graph lambda
on the depth of learning necessary. Rationale in both is that the "meaning" is accumulated left-right and top-down or

---

293.(FEATURE-DONE) Commits - 7 July 2016 - Experimental implementation of Recursive Lambda Function Growth Algorithm

---

1.Each Sentence Text is converted to an AVL Balanced Tree (inorder traversal of this tree creates the original text)
2.Postorder traversal of this tree computes a postfix expression
3.Postfix expression is evaluated and a lambda function composition is generated by parenthesization denoting a funct
for this have been committed to testlogs/ which show the inorder and postorder traversal and the final lambda functic
4.This is different from Part of Speech tree or a Context Free Grammar parse tree.
5.On an average every english sentence has a connective on every third word which is exactly what inorder traversal c

---

294. (FEATURE-DONE) Commits - 8 July 2016

Changed Postfix Evaluation to Infix evaluation in Lambda Function Growth with logs in testlogs/

---

295. (THEORY) Contextual Multi-Armed Bandits, Reinforcement Learning and ThoughtNet - related to 241, 291, 292 - 18 J

Contextual Multi-Armed Bandits are the class of problems where a choice has to be made amongst k arms and each iterat

Reference:

295.1 Multiword Testing Decision Service - http:*//arxiv.org/pdf/1606.03966v1.pdf*

---

296. (FEATURE-DONE) Music Pattern Mining - Jensen-Shannon Divergence Distance between two FFT Frequency Distributions

---

Commits - 20 July 2016 - related to 68, 69

(**NO**TE: Because of some weird SourceForge/GitHub error, FFT txt files added to repos on 19 July 2016 are still flagged
added to repos again along with new files)

FFTs of 2 audio files are parsed and written to *_trimmed.txt by awk to contain only the frequencies for each sample.
These files are read by JensenShannonDivergence.py to compute the JS Distance between these two FFT frequency distrib
Preprocessing is done so as to normalize the frequency to map to a probability (frequency/sum_of_frequencies) which g
probability distribution from frequency distribution. Jensen-Shannon Distance which is the weighted average of bidire
Kullback-Leibler Divergence measures of the two distributions, indicates similarity or distance between two music sam
quantitatively.

There are 4 music samples: music_pattern_mining/FFT_classical_1_20July2016.txt and music_pattern_mining/FFT_classical
are similar (similar notes sung by different musicians). music_pattern_mining/FFT_classical_1_19July2016.txt and
music_pattern_mining/FFT_classical_2_19July2016.txt are also similar (different set of notes sung by different musici
distance across these 4 ordered pairs is captured and committed in testlogs/

Jensen-Shannon Distance across 2 FFT Frquency-Probability distributions of music samples is a simple, basic measure f
basis for clustering and classification of music. There could be some inaccuracies because Audacity does not generate
music file but only for first ~5 minutes. Two similar notes with different musicians could be distant and two dissimi
close because of this Audacity limitation. Presently noise filtering and cherrypick peak frequencies is not done and
entire frequency range is compared.

---

297. (FEATURE-DONE) Software Analytics - Cyclomatic Complexity from SATURN .dot graphs - related to 65

Commits - 22 July 2016

New Python-Spark implementation that reads the SATURN program analyzer generated .dot graph files processes them with
MapReduce to find the number of edges and vertices in the graph represented by .dot file for each snippet. From this
Cyclomatic Complexity is calculated (which is Edges - Vertices + 2) - a standard Function Point Estimator for code c
There were some JVM crashes frequently while starting up spark-submit logs for which are also committed in testlogs a
the two successful Spark computed Cyclomatic Complexity measures for two .dot files. These .dot files are sourced fro
VIRGO Linux saturn program analysis kernel driver.

---

298. (FEATURE-DONE) String Search - Longest Common Substring - Suffix Trees(Suffix Arrays+LCP) Implementation

```
4756 ----------------------------------------------------------------------------------------------------
4757 Commits - 26 July 2016
4758 ----------------------------------------------------------------------------------------------------
4759 This implementation finds the most repeated substrings within a larger string by constructing Suffix Trees
4760 indirectly with Suffix Array and Longest Common Prefix (LCP) datastructures. An ideal application for String Search i
4761 in Bioinformatics, Streamed Data analysis etc., For example, a Time Series data with fluctuating curve can be
4762 encoded as a huge binary string by mapping ebb and tide to 0 and 1. Thus a function graph time series is projected on
4763 {0,1} alphabet to create blob. Usual Time Series analysis mines for Trends, Cycles, Seasons and Irregularities over t
4764 This binary encoding of time series gives an alternative spectacle to look at the trends (highs and lows). Longest re
4765 pattern in this binary encoding is a cycle. Suffix Array by [UdiManber] has been implemented over Suffix Trees implem
4766 of [Weiner] and [Ukkonen] because of simplicity of Suffix Arrays and Suffix Trees=Suffix Arrays + LCPs.
4767
4768 ----------------------------------------------------------------------------------------------------
4769 299. (FEATURE-DONE) Binary String Encoding of Time Series - Commits - 27 July 2016
4770 ----------------------------------------------------------------------------------------------------
4771 1. New python script to encode a time series datastream as binary string has been added to repository.
4772 2. This writes to StringSearch_Pattern.txt which is read by StringSearch_LongestRepeatedSubstring.py
4773 3. Encoding scheme: If next data point is smaller, write 0 else if greater write 1. This captures the
4774 fluctuations in dataset irrespective of the amplitudes at each point.
4775 4. For example a bitonic sequence would have been encoded as ....111111110000000.... (ascends and descends)
4776 5. Suffix Array + LCP algorithm on this sequence finds the Longest Repeated Substring. For a specific example of
4777 Stock ticker time series data this amounts to frequently recurring fluctuation pattern.
4778 6. Logs for the above have been committed to testlogs/
4779 7. Every time series dataset is a union of bitonic sequences with varying lengths corresponding to peaks and troughs
4780 and has self-similar fractal structure (zooming out the series has similarities to original series) . This implies th
4781 binary string encoded as previously is fractal too.
4782
4783 ----------------------------------------------------------------------------------------------------
4784 300. (FEATURE-DONE) Tornado GUI Authentication Template - Commits - 27 July 2016
4785 ----------------------------------------------------------------------------------------------------
4786 Rewritten Tornado GUI with Login Template and redirect to Algorithms Execution Template. Presently implemented for on
4787 root user by cookie setting. Entry point has been changed to http://host:33333/neuronrain_auth (Login).
4788
4789 ----------------------------------------------------------------------------------------------------
4790 301. (FEATURE-DONE) OAuth2 authentication and credentials storage in MongoDB/Redis with passlib sha256 encryption
4791 ----------------------------------------------------------------------------------------------------
4792 1.Login Handler has been augmented with OAuth2 authentication by python-oauth2 library.
4793 2.MongoDB is used as OAuth2 credentials storage backend and Redis for token storage backend
4794 3.Passlib is used to encrypt the password with SHA256 hash digest and stored once in MongoDB (code is commented)
4795 4.Login page password is verified with passlib generated hash in MongoDB queried for a username argument
4796 5.With this minimal standard authentication has been implemented for NeuronRain cloud deployment.
4797 6.Prerequisite is to populate MongoDB neuronrain_oauth database and neuronrain_users collections with
4798 passlib encypted JSON by uncommenting the collections.insert_one() code.
4799
4800 ----------------------------------------------------------------------------------------------------
4801 302. Commits - 29 July 2016 - boost::python extensions for VIRGO Linux Kernel system calls
4802 ----------------------------------------------------------------------------------------------------
4803 virgo_clone() system call has been included in switch-case and invokes an exported kernel module function in kernelsp
4804
4805 ----------------------------------------------------------------------------------------------------
4806 303. (THEORY) Update on Bitonic Sort Merge Tree for Discrete Hyperbolic Factorization Spark Cloud Implementation in 3
4807 ----------------------------------------------------------------------------------------------------
4808 Bitonic Sort Merge Tree for Discrete Hyperbolic Factorization described in http://sourceforge.net/projects/acadpdraft
4809
4810 303.1 Bitonic Sort requires O((logN)^2) time with O(N*(logN)^2) processors.
4811 303.2 Discretized Hyperbolic arc of length N can be split into N/logN segments of length logN each.
4812 303.3 Above N/logN segments can be grouped in leaf level of merge tree into N/2logN pairs of (logN,logN) = 2logN leng
4813 303.4 Each such pair can be sorted in log(2logN) time in parallel with N/(2logN) * N(logN)^2 = N^2/2logN comparator p
4814 303.5 Height of the merge tree is O(log(N/logN)). Total sort time is sum over sorts at each level of merge tree = log
4815 303.6 Total sort time of merge tree is upperbounded as (not tight one) height * maximum_sort_time_per_level:
4816         <= O(log(N/logN) * log(N/logN*logN)) = O(log(N/logN) * log(N))
4817       with maximum of N^2/2logN processor comparators at leaf which tapers down up the tree.
4818 303.7 Binary Search on final sorted merged tile requires additional O(logN) which effectively reduces to:
4819         O(log(N/logN)*logN + logN) = O(log(N/logN)*logN) <= O((logN)^2) time
4820       with N^2/2logN processor comparators for finding a factor.
4821 303.8 This is comparably better bound than the Parallel RAM ANSV algorithm based merge sort alternatives and easy to
4822 303.9 Again the input size is N and not logN, but yet definition of NC is abided by. It remains an open question whet
4823
4824 ----------------------------------------------------------------------------------------------------
4825 304. Commits - 31 July 2016 - boost::python C++ and cpython virgo_clone() system call invocations
4826 ----------------------------------------------------------------------------------------------------
4827 1. Boost C++ Python extensions - virgo_clone() system call has been included in switch-case for cpython invocation of
4828 2. CPython extensions - test log for boost::python virgo_clone() invocation with use_as_kingcobra_service=0 which dir
4829
4830 ----------------------------------------------------------------------------------------------------
4831 305. NEURONRAIN VIRGO Commits for ASFER Boost::Python C++ virgo memory system calls invocations
4832 ----------------------------------------------------------------------------------------------------
4833 (BUG - STABILITY ISSUES) Commits - 1 August 2016 - VIRGO Linux Stability Issues - Ongoing Random oops and panics inve
4834 ----------------------------------------------------------------------------------------------------
4835 1. GFP_KERNEL has been replaced with GFP_ATOMIC flags in kmem allocations.
4836 2. NULL checks have been introduced in lot of places involving strcpy, strcat, strcmp etc., to circumvent
```

```
buffer overflows.
3. Though this has stabilized the driver to some extent, still there are OOPS in unrelated places deep
with in kernel where paging datastructures are accessed - kmalloc somehow corrupts paging
4. OOPS are debugged via gdb as:
        4.1 gdb ./vmlinux /proc/kcore
        or
        4.2 gdb <loadable_kernel_module>.o
   followed by
        4.3 l *(address+offset in OOPS dump)
5. kern.log(s) for the above have been committed in tar.gz format and have numerous OOPS occurred during repetitive t
invocation(boost::python C++) invocations of virgo memory system calls.
6. Paging related OOPS look like an offshoot of set_fs() encompassing the filp_open VFS calls.
7. In C++ Boost::Python extensions, flag changed for VIRGO memory system calls invocation from python.

-----------------------------------------------------------------------------------------------------------------
306. Commits - 3 August 2016
-----------------------------------------------------------------------------------------------------------------
Social Network Analysis with Twitter - Changes to Belief Propagation Potential Computation
-----------------------------------------------------------------------------------------------------------------
1. Exception handling for UnicodeError has been added in SentimentAnalyzer
2. Belief Propagation Potential computation for the RGO graph constructed has been changed to do plain
summation of positivity and negativity scores for DFS of K-Core rather than multiplication which heuristically
appears to predict sentiments better
3. An example for tweets sentiments analysis for 2 search keywords has been logged and committed in testlogs/
-------------------------------------------------------------------------------------
4. Excerpts for sentiment scores - positive and negative - from RGO graph of few tweets
-------------------------------------------------------------------------------------
root@shrinivaasanka-Inspiron-1545:/media/shrinivaasanka/0fc4d8a2-1c74-42b8-8099-9ef78d8c8ea2/home/kashrinivaasan/Kris
K-Core DFS belief_propagated_posscore: 6.078125
K-Core DFS belief_propagated_negscore: 8.140625
Core Number belief_propagated_posscore: 25.125
Core Number belief_propagated_negscore: 26.125
K-Core DFS belief_propagated_posscore: 6.078125
K-Core DFS belief_propagated_negscore: 8.140625
Core Number belief_propagated_posscore: 25.125
Core Number belief_propagated_negscore: 26.125
K-Core DFS belief_propagated_posscore: 55.09375
K-Core DFS belief_propagated_negscore: 54.3125
Core Number belief_propagated_posscore: 92.625
Core Number belief_propagated_negscore: 93.875
K-Core DFS belief_propagated_posscore: 60.09375
K-Core DFS belief_propagated_negscore: 59.3125
Core Number belief_propagated_posscore: 118.625
Core Number belief_propagated_negscore: 117.75
root@shrinivaasanka-Inspiron-1545:/media/shrinivaasanka/0fc4d8a2-1c74-42b8-8099-9ef78d8c8ea2/home/kashrinivaasan/Kris
K-Core DFS belief_propagated_posscore: 34.0
K-Core DFS belief_propagated_negscore: 30.359375
Core Number belief_propagated_posscore: 90.25
Core Number belief_propagated_negscore: 85.125
K-Core DFS belief_propagated_posscore: 54.0
K-Core DFS belief_propagated_negscore: 46.734375
Core Number belief_propagated_posscore: 111.125
Core Number belief_propagated_negscore: 108.125
K-Core DFS belief_propagated_posscore: 97.203125
K-Core DFS belief_propagated_negscore: 91.015625
Core Number belief_propagated_posscore: 172.125
Core Number belief_propagated_negscore: 172.125
K-Core DFS belief_propagated_posscore: 97.203125
K-Core DFS belief_propagated_negscore: 91.015625
Core Number belief_propagated_posscore: 178.125
Core Number belief_propagated_negscore: 178.125

5. Majority Predominant Sentiment indicated by above scores for randomly sampled tweets can be used as one of the Ele

-----------------------------------------------------------------------------------------------------------------
307. (THEORY) Sentiment Analysis from Recursive Gloss Overlap as a Voting Function in Majority Voting - related to 14
-----------------------------------------------------------------------------------------------------------------
As mentioned in 306, Sentiment Analysis scores derived from Recursive Gloss Overlap graph of a text is an indirect vo
overall sentiment is negative it is an "against" vote and when positive it is a "for" vote. In this sense, Sentiment

-----------------------------------------------------------------------------------------------------------------
308. (FEATURE-DONE) Markov Random Fields (MRF) Belief Propagation - Commits - 8 August 2016
-----------------------------------------------------------------------------------------------------------------
Existing Sentiment Analyzer does belief propagation by computing product of potentials in DFS traversal only.
New function for Markov Random Fields Belief Propagation has been included which handles the generalized case of beli
graph by factoring it into maximal cliques and finding potentials per clique. These per clique potentials are multipl
with number of cliques to get polarized sentiments scores - positive, negative and objective. Logs for Sentiment Anal
with MRF have been committed to testlogs/

Reference:
---------
```

308.1 Introduction to Machine Learning - [Ethem Alpaydin] - Graphical Models

---

309. (FEATURE-DONE) Commits - 12 August 2016 - Boost 1.58.0 upgrade from 1.55.0 - Boost::Python C++ extensions for VI

1.Rebuilt Boost::Python extensions for VIRGO Linux kernel system calls with boost 1.58 upgraded from boost 1.55.
2.kern.log for miscellaneous VIRGO system calls from both telnet and system call request routes has been compressed a
testlogs (~300MB).
Following multiple debug options were tried out for heisencrash resolution (Note to self):
3./etc/sysctl.conf has been updated with kernel panic tunables with which the mean time between crashes has increased
deep within kernel (VMA paging) - commits for this are in VIRGO linux tree. With these settings following usecase wor
        virgo_cloud_malloc()
        virgo_cloud_set()
        virgo_cloud_get()
        virgo_cloud_set() overwrite
        virgo_cloud_get()
through telnet route.
4.Debugging VIRGO linux kernel with an Oracle VirtualBox virtual machine installation and debugging the VM with a net
5.Debugging VIRGO linux kernel with QEMU-KVM installation and debugging the VM with a netconsole port via KGDB is bei
6.Path to boost::python extension .so has been updated.

---

310. (THEORY) DFT of a Sliding Window Fragment of Time Series, Integer Partitions and Hashing

---

Time Series Data Stream can be viewed through a sliding window of some fixed width moving across the data. Captured d
window are analyzed in frequency domain with a Discrete Fourier Transform. If there are n frequencies in the window c
sinusoids which superimpose to form the original data. In discrete sense, sinusoids partition the data at any time po
relation between hash functions and integer partitions in https://*sites.google.com/site/kuja27/IntegerPartitionAndHas*
each of n discrete sinusoid in DFT corresponds to a hash function and the time-series sliding window is an amalgamati

---

311. (THEORY) Pr(Good) Majority Voting Circuit, Percolation, PRG choice and Boolean Circuit Composition - related to

---

Even though Percolation circuit is in Noise Stability Regime with zero sensitivity and 100% stability, delta term mer
of 14 and 53 (second column) need not be zero which could force percolation circuit to be in BPNC. Problem is then de

References:

311.1 Mathematical Techniques for Analysis of Boolean Functions - [AnnaBernasconi] - 2.8.2 Boolean Function Compositi
311.2 Properties and Applications of Boolean Function Composition - [AvishayTal] - eccc.hpi-web.de/report/2012/163/dc
311.3 Boolean Circuit Composition - www.cs.tau.ac.il/~safra/ACT/CCandSpaceB.ppt

---

312. (THEORY) Circuit SAT lowerbounds for Pr(Good) Majority Voting - related 14 and 129

312.1 RHS of Pr(Good) Majority Voting circuit is an NC circuit in composition with CircuitSAT for each voter. This pa
requires some kind of barrier synchronization so that input from all voters is available to NC Majority Circuit. This
to be ignored because the barrier delay depends on the voter whose decision time is the most. If in worst case the bo
in EXP then barrier synchronization is also in EXP. It is also assumed that voters decide independent of each other.
312.2 CircuitSAT is the circuit version of CNF SAT concerned with the bounds for finding satisfying assignments to a
312.3 An arbitrary Circuit's Satisfiability requires O(2^0.4058m) where m is the number of gates in Circuit(i.e. size
312.4 Because of the fact that Pr(good) majority voting is a circuit composition with voter circuit SATs as inputs to
312.5 Let number of voters be n and number of gates on the average per voter be m. NC1 majority requires O(logn) time
312.6 If number of voters is exponential in number of gates per voter then n=2^m which is the worst case scenario and
312.7 In a very generic case, as mentioned in 275, each voter can have a non-boolean voting function viz., a linear p
        312.7.1 NC^P (if voting functions are real LP) and LHS NC/poly or BPNC percolation lowerbounds NC^P when both
        312.7.2 NC^NP (if voting functions are integer LP) and LHS NC/poly or BPNC percolation lowerbounds NC^NP when
312.8 Circuit Size of Pr(Good) RHS is non-trivial to determine. All that is known so far is the bound for CircuitSAT
312.9 Assumption 1: Each voter has different variables - set of variables of voters are all pairwise disjoint.
312.10 If number of voters is arbitrarily huge and if number of voters is exponential in number of gates per voter SA
312.11 Even if m (number of gates per voter SAT) is assumed to be polynomial in number of input variables i.e m=f(v)
312.12 If both LHS and RHS of Pr(Good) are of zero or equal error, LHS lowerbounds RHS (an assumption made throughout
312.13 Important notion in representing a voter boolean function by a circuit is Hardness which is defined as:
A boolean function f:{0,1}^n->{0,1} is h-hard if there exists a circuit C of size s such that for all x in {0,1}^n, P
312.14 There are two classes of errors: Voter error + Voting error. Voter error is the Noise Sensitivity of a voter b

References:

312.15 Complexity Hardness of Noisy Boolean Functions - http://*cstheory.stackexchange.com/questions/18822/hardness-of*
312.16 Hardness of Boolean Functions and Amplification - [RyanODonnell] - https://*www.cs.cmu.edu/~odonnell/boolean-ar*
312.17 Probabilistic Boolean Logic - [LakshmiNBChakrapani-KrishnaPalem] - http://*www.ece.rice.edu/~al4/visen/2008rice*
312.18 How much a Linear Programming Oracle speedup Polytime algorithms - http://*cstheory.stackexchange.com/questions*
312.19 Algorithms for Circuits and Circuits for Algorithms - [RyanWilliams] - http://*web.stanford.edu/~rrwill/ICM-sur*
312.20 Depth 3 multilinear circuits - [OdedGoldreich] - http://*www.wisdom.weizmann.ac.il/~oded/R3/kk.pdf - D-canonica*
312.21 Size hierarchy theorem for circuits - http://*cstheory.stackexchange.com/questions/5110/hierarchy-theorem-for-c*
312.22 Size Hierarchy [Lupanov] - Number of functions computable by circuit of size s - Lemma 2.1 counting argument -
312.23 Efficient Parallel Computation = NC - [AroraBarak] - http://*theory.cs.princeton.edu/complexity/book.pdf - Theo*
312.24 Size hierarchy of 312.22 in Pr(Good) majority voting circuit context implies that number of functions computab
312.25 If circuit sizes differ in zero-error case on both sides i.e when 100% convergence occurs (or) equal error on
312.26 There is an alternative definition of Boolean function hardness in http://*www.math.ias.edu/~avi/PUBLICATIONS/M*

```
4999  ----------------------------------------------------------------------------------------------------------------
5000  313. (FEATURE-DONE) Astronomical Sequence Mining based Precipitation Forecast (from 100 year historic dataset) - rela
5001  ----------------------------------------------------------------------------------------------------------------
5002  Following the earlier experimental pattern mining done on astronomical datasets and their striking correlational coin
5003
5004  python MaitreyaEncHoro_RuleSearch.py  --min_year=2016 --min_month=11 --min_days=15 --min_hours=10 --min_minutes=10 --
5005
5006  {  --date="2016-11-29 10:10:10 5"  --location=" x 77:0:0 7:0:0 " --planet-list } - There is a Class Association Rule
5007  {  --date="2016-11-30 10:10:10 5"  --location=" x 77:0:0 7:0:0 " --planet-list } - There is a Class Association Rule
5008  {  --date="2016-12-1 10:10:10 5"  --location=" x 77:0:0 7:0:0 " --planet-list } - There is a Class Association Rule m
5009  {  --date="2016-12-2 10:10:10 5"  --location=" x 77:0:0 7:0:0 " --planet-list } - There is a Class Association Rule m
5010  {  --date="2016-11-29 10:10:10 5"  --location=" x 77:0:0 7:1:0 " --planet-list } - There is a Class Association Rule
5011  {  --date="2016-11-30 10:10:10 5"  --location=" x 77:0:0 7:1:0 " --planet-list } - There is a Class Association Rule
5012  {  --date="2016-12-1 10:10:10 5"  --location=" x 77:0:0 7:1:0 " --planet-list } - There is a Class Association Rule m
5013  {  --date="2016-12-2 10:10:10 5"  --location=" x 77:0:0 7:1:0 " --planet-list } - There is a Class Association Rule m
5014  {  --date="2016-11-29 10:10:10 5"  --location=" x 77:0:0 7:2:0 " --planet-list } - There is a Class Association Rule
5015  {  --date="2016-11-30 10:10:10 5"  --location=" x 77:0:0 7:2:0 " --planet-list } - There is a Class Association Rule
5016  {  --date="2016-12-1 10:10:10 5"  --location=" x 77:0:0 7:2:0 " --planet-list } - There is a Class Association Rule m
5017  {  --date="2016-12-2 10:10:10 5"  --location=" x 77:0:0 7:2:0 " --planet-list } - There is a Class Association Rule m
5018  {  --date="2016-11-29 10:10:10 5"  --location=" x 77:0:0 7:3:0 " --planet-list } - There is a Class Association Rule
5019  {  --date="2016-11-30 10:10:10 5"  --location=" x 77:0:0 7:3:0 " --planet-list } - There is a Class Association Rule
5020  {  --date="2016-12-1 10:10:10 5"  --location=" x 77:0:0 7:3:0 " --planet-list } - There is a Class Association Rule m
5021  {  --date="2016-12-2 10:10:10 5"  --location=" x 77:0:0 7:3:0 " --planet-list } - There is a Class Association Rule m
5022  {  --date="2016-11-29 10:10:10 5"  --location=" x 77:0:0 7:4:0 " --planet-list } - There is a Class Association Rule
5023  {  --date="2016-11-30 10:10:10 5"  --location=" x 77:0:0 7:4:0 " --planet-list } - There is a Class Association Rule
5024  {  --date="2016-12-1 10:10:10 5"  --location=" x 77:0:0 7:4:0 " --planet-list } - There is a Class Association Rule m
5025  {  --date="2016-12-2 10:10:10 5"  --location=" x 77:0:0 7:4:0 " --planet-list } - There is a Class Association Rule m
5026  {  --date="2016-11-29 10:10:10 5"  --location=" x 77:0:0 7:5:0 " --planet-list } - There is a Class Association Rule
5027  {  --date="2016-11-30 10:10:10 5"  --location=" x 77:0:0 7:5:0 " --planet-list } - There is a Class Association Rule
5028  {  --date="2016-12-1 10:10:10 5"  --location=" x 77:0:0 7:5:0 " --planet-list } - There is a Class Association Rule m
5029  {  --date="2016-12-2 10:10:10 5"  --location=" x 77:0:0 7:5:0 " --planet-list } - There is a Class Association Rule m
5030  {  --date="2016-11-29 10:10:10 5"  --location=" x 77:0:0 7:6:0 " --planet-list } - There is a Class Association Rule
5031  {  --date="2016-11-30 10:10:10 5"  --location=" x 77:0:0 7:6:0 " --planet-list } - There is a Class Association Rule
5032  {  --date="2016-12-1 10:10:10 5"  --location=" x 77:0:0 7:6:0 " --planet-list } - There is a Class Association Rule m
5033  {  --date="2016-12-2 10:10:10 5"  --location=" x 77:0:0 7:6:0 " --planet-list } - There is a Class Association Rule m
5034  {  --date="2016-11-29 10:10:10 5"  --location=" x 77:0:0 7:7:0 " --planet-list } - There is a Class Association Rule
5035  {  --date="2016-11-30 10:10:10 5"  --location=" x 77:0:0 7:7:0 " --planet-list } - There is a Class Association Rule
5036  {  --date="2016-12-1 10:10:10 5"  --location=" x 77:0:0 7:7:0 " --planet-list } - There is a Class Association Rule m
5037  {  --date="2016-12-2 10:10:10 5"  --location=" x 77:0:0 7:7:0 " --planet-list } - There is a Class Association Rule m
5038  {  --date="2016-11-29 10:10:10 5"  --location=" x 77:0:0 7:8:0 " --planet-list } - There is a Class Association Rule
5039  {  --date="2016-11-30 10:10:10 5"  --location=" x 77:0:0 7:8:0 " --planet-list } - There is a Class Association Rule
5040  {  --date="2016-12-1 10:10:10 5"  --location=" x 77:0:0 7:8:0 " --planet-list } - There is a Class Association Rule m
5041  {  --date="2016-12-2 10:10:10 5"  --location=" x 77:0:0 7:8:0 " --planet-list } - There is a Class Association Rule m
5042  {  --date="2016-11-29 10:10:10 5"  --location=" x 77:0:0 8:0:0 " --planet-list } - There is a Class Association Rule
5043  {  --date="2016-11-30 10:10:10 5"  --location=" x 77:0:0 8:0:0 " --planet-list } - There is a Class Association Rule
5044  {  --date="2016-12-1 10:10:10 5"  --location=" x 77:0:0 8:0:0 " --planet-list } - There is a Class Association Rule m
5045  {  --date="2016-12-2 10:10:10 5"  --location=" x 77:0:0 8:0:0 " --planet-list } - There is a Class Association Rule m
5046  {  --date="2016-11-29 10:10:10 5"  --location=" x 77:0:0 8:1:0 " --planet-list } - There is a Class Association Rule
5047  {  --date="2016-11-30 10:10:10 5"  --location=" x 77:0:0 8:1:0 " --planet-list } - There is a Class Association Rule
5048  {  --date="2016-12-1 10:10:10 5"  --location=" x 77:0:0 8:1:0 " --planet-list } - There is a Class Association Rule m
5049  {  --date="2016-12-2 10:10:10 5"  --location=" x 77:0:0 8:1:0 " --planet-list } - There is a Class Association Rule m
5050  {  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:0:0 8:2:0 " --planet-list } - There is a Class Association Rul
5051  {  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:0:0 8:2:0 " --planet-list } - There is a Class Association Rul
5052  {  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:0:0 8:2:0 " --planet-list } - There is a Class Association Rule
5053  {  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:0:0 8:2:0 " --planet-list } - There is a Class Association Rule
5054  {  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:0:0 8:3:0 " --planet-list } - There is a Class Association Rul
5055  {  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:0:0 8:3:0 " --planet-list } - There is a Class Association Rul
5056  {  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:0:0 8:3:0 " --planet-list } - There is a Class Association Rule
5057  {  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:0:0 8:3:0 " --planet-list } - There is a Class Association Rule
5058  {  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:0:0 8:4:0 " --planet-list } - There is a Class Association Rul
5059  {  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:0:0 8:4:0 " --planet-list } - There is a Class Association Rul
5060  {  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:0:0 8:4:0 " --planet-list } - There is a Class Association Rule
5061  {  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:0:0 8:4:0 " --planet-list } - There is a Class Association Rule
5062  {  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:0:0 8:5:0 " --planet-list } - There is a Class Association Rul
5063  {  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:0:0 8:5:0 " --planet-list } - There is a Class Association Rul
5064  {  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:0:0 8:5:0 " --planet-list } - There is a Class Association Rule
5065  {  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:0:0 8:5:0 " --planet-list } - There is a Class Association Rule
5066  {  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:0:0 8:6:0 " --planet-list } - There is a Class Association Rul
5067  {  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:0:0 8:6:0 " --planet-list } - There is a Class Association Rul
5068  {  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:0:0 8:6:0 " --planet-list } - There is a Class Association Rule
5069  {  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:0:0 8:6:0 " --planet-list } - There is a Class Association Rule
5070  {  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:0:0 8:7:0 " --planet-list } - There is a Class Association Rul
5071  {  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:0:0 8:7:0 " --planet-list } - There is a Class Association Rul
5072  {  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:0:0 8:7:0 " --planet-list } - There is a Class Association Rule
5073  {  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:0:0 8:7:0 " --planet-list } - There is a Class Association Rule
5074  {  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:0:0 8:8:0 " --planet-list } - There is a Class Association Rul
5075  {  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:0:0 8:8:0 " --planet-list } - There is a Class Association Rul
5076  {  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:0:0 8:8:0 " --planet-list } - There is a Class Association Rule
5077  {  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:0:0 8:8:0 " --planet-list } - There is a Class Association Rule
5078  {  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:0:0 9:0:0 " --planet-list } - There is a Class Association Rul
5079  {  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:0:0 9:0:0 " --planet-list } - There is a Class Association Rul
```

```
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:0:0 9:0:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:0:0 9:0:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:0:0 9:1:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:0:0 9:1:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:0:0 9:1:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:0:0 9:1:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:0:0 9:2:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:0:0 9:2:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:0:0 9:2:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:0:0 9:2:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:0:0 9:3:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:0:0 9:3:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:0:0 9:3:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:0:0 9:3:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:0:0 9:4:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:0:0 9:4:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:0:0 9:4:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:0:0 9:4:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:0:0 9:5:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:0:0 9:5:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:0:0 9:5:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:0:0 9:5:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:0:0 9:6:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:0:0 9:6:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:0:0 9:6:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:0:0 9:6:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:0:0 9:7:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:0:0 9:7:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:0:0 9:7:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:0:0 9:7:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:0:0 9:8:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:0:0 9:8:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:0:0 9:8:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:0:0 9:8:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5"  --location=" x 77:1:0 7:0:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-30 10:10:10 5"  --location=" x 77:1:0 7:0:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-1 10:10:10 5"  --location=" x 77:1:0 7:0:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-12-2 10:10:10 5"  --location=" x 77:1:0 7:0:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-11-29 10:10:10 5"  --location=" x 77:1:0 7:1:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-30 10:10:10 5"  --location=" x 77:1:0 7:1:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-1 10:10:10 5"  --location=" x 77:1:0 7:1:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-12-2 10:10:10 5"  --location=" x 77:1:0 7:1:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-11-29 10:10:10 5"  --location=" x 77:1:0 7:2:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-30 10:10:10 5"  --location=" x 77:1:0 7:2:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-1 10:10:10 5"  --location=" x 77:1:0 7:2:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-12-2 10:10:10 5"  --location=" x 77:1:0 7:2:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-11-29 10:10:10 5"  --location=" x 77:1:0 7:3:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-30 10:10:10 5"  --location=" x 77:1:0 7:3:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-1 10:10:10 5"  --location=" x 77:1:0 7:3:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-12-2 10:10:10 5"  --location=" x 77:1:0 7:3:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-11-29 10:10:10 5"  --location=" x 77:1:0 7:4:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-30 10:10:10 5"  --location=" x 77:1:0 7:4:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-1 10:10:10 5"  --location=" x 77:1:0 7:4:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-12-2 10:10:10 5"  --location=" x 77:1:0 7:4:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-11-29 10:10:10 5"  --location=" x 77:1:0 7:5:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-30 10:10:10 5"  --location=" x 77:1:0 7:5:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-1 10:10:10 5"  --location=" x 77:1:0 7:5:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-12-2 10:10:10 5"  --location=" x 77:1:0 7:5:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-11-29 10:10:10 5"  --location=" x 77:1:0 7:6:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-30 10:10:10 5"  --location=" x 77:1:0 7:6:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-1 10:10:10 5"  --location=" x 77:1:0 7:6:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-12-2 10:10:10 5"  --location=" x 77:1:0 7:6:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-11-29 10:10:10 5"  --location=" x 77:1:0 7:7:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-30 10:10:10 5"  --location=" x 77:1:0 7:7:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-1 10:10:10 5"  --location=" x 77:1:0 7:7:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-12-2 10:10:10 5"  --location=" x 77:1:0 7:7:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-11-29 10:10:10 5"  --location=" x 77:1:0 7:8:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-30 10:10:10 5"  --location=" x 77:1:0 7:8:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-1 10:10:10 5"  --location=" x 77:1:0 7:8:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-12-2 10:10:10 5"  --location=" x 77:1:0 7:8:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-11-29 10:10:10 5"  --location=" x 77:1:0 8:0:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-30 10:10:10 5"  --location=" x 77:1:0 8:0:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-1 10:10:10 5"  --location=" x 77:1:0 8:0:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-12-2 10:10:10 5"  --location=" x 77:1:0 8:0:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:1:0 8:1:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:1:0 8:1:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:1:0 8:1:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:1:0 8:1:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:1:0 8:2:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:1:0 8:2:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:1:0 8:2:0 " --planet-list } - There is a Class Association Rule
```

```
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:1:0 8:2:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:1:0 8:3:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:1:0 8:3:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:1:0 8:3:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:1:0 8:3:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:1:0 8:4:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:1:0 8:4:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:1:0 8:4:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:1:0 8:4:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:1:0 8:5:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:1:0 8:5:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:1:0 8:5:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:1:0 8:5:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:1:0 8:6:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:1:0 8:6:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:1:0 8:6:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:1:0 8:6:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:1:0 8:7:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:1:0 8:7:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:1:0 8:7:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:1:0 8:7:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:1:0 8:8:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:1:0 8:8:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:1:0 8:8:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:1:0 8:8:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:1:0 9:0:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:1:0 9:0:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:1:0 9:0:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:1:0 9:0:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:1:0 9:1:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:1:0 9:1:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:1:0 9:1:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:1:0 9:1:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:1:0 9:2:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:1:0 9:2:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:1:0 9:2:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:1:0 9:2:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:1:0 9:3:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:1:0 9:3:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:1:0 9:3:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:1:0 9:3:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:1:0 9:4:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:1:0 9:4:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:1:0 9:4:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:1:0 9:4:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:1:0 9:5:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:1:0 9:5:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:1:0 9:5:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:1:0 9:5:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:1:0 9:6:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:1:0 9:6:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:1:0 9:6:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:1:0 9:6:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:1:0 9:7:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:1:0 9:7:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:1:0 9:7:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:1:0 9:7:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:1:0 9:8:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:1:0 9:8:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:1:0 9:8:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:1:0 9:8:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5"  --location=" x 77:2:0 7:0:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-30 10:10:10 5"  --location=" x 77:2:0 7:0:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-1 10:10:10 5"  --location=" x 77:2:0 7:0:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-12-2 10:10:10 5"  --location=" x 77:2:0 7:0:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-11-29 10:10:10 5"  --location=" x 77:2:0 7:1:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-30 10:10:10 5"  --location=" x 77:2:0 7:1:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-1 10:10:10 5"  --location=" x 77:2:0 7:1:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-12-2 10:10:10 5"  --location=" x 77:2:0 7:1:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-11-29 10:10:10 5"  --location=" x 77:2:0 7:2:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-30 10:10:10 5"  --location=" x 77:2:0 7:2:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-1 10:10:10 5"  --location=" x 77:2:0 7:2:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-12-2 10:10:10 5"  --location=" x 77:2:0 7:2:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-11-29 10:10:10 5"  --location=" x 77:2:0 7:3:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-30 10:10:10 5"  --location=" x 77:2:0 7:3:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-1 10:10:10 5"  --location=" x 77:2:0 7:3:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-12-2 10:10:10 5"  --location=" x 77:2:0 7:3:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-11-29 10:10:10 5"  --location=" x 77:2:0 7:4:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-30 10:10:10 5"  --location=" x 77:2:0 7:4:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-1 10:10:10 5"  --location=" x 77:2:0 7:4:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-12-2 10:10:10 5"  --location=" x 77:2:0 7:4:0 " --planet-list } - There is a Class Association Rule m
```

```
{  --date="2016-11-29 10:10:10 5"  --location=" x 77:2:0 7:5:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-30 10:10:10 5"  --location=" x 77:2:0 7:5:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-1 10:10:10 5"  --location=" x 77:2:0 7:5:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-12-2 10:10:10 5"  --location=" x 77:2:0 7:5:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-11-29 10:10:10 5"  --location=" x 77:2:0 7:6:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-30 10:10:10 5"  --location=" x 77:2:0 7:6:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-1 10:10:10 5"  --location=" x 77:2:0 7:6:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-12-2 10:10:10 5"  --location=" x 77:2:0 7:6:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-11-29 10:10:10 5"  --location=" x 77:2:0 7:7:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-30 10:10:10 5"  --location=" x 77:2:0 7:7:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-1 10:10:10 5"  --location=" x 77:2:0 7:7:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-12-2 10:10:10 5"  --location=" x 77:2:0 7:7:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-11-29 10:10:10 5"  --location=" x 77:2:0 7:8:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-30 10:10:10 5"  --location=" x 77:2:0 7:8:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-1 10:10:10 5"  --location=" x 77:2:0 7:8:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-12-2 10:10:10 5"  --location=" x 77:2:0 7:8:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:2:0 8:0:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:2:0 8:0:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:2:0 8:0:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:2:0 8:0:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:2:0 8:1:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:2:0 8:1:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:2:0 8:1:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:2:0 8:1:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:2:0 8:2:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:2:0 8:2:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:2:0 8:2:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:2:0 8:2:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:2:0 8:3:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:2:0 8:3:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:2:0 8:3:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:2:0 8:3:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:2:0 8:4:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:2:0 8:4:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:2:0 8:4:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:2:0 8:4:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:2:0 8:5:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:2:0 8:5:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:2:0 8:5:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:2:0 8:5:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:2:0 8:6:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:2:0 8:6:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:2:0 8:6:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:2:0 8:6:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:2:0 8:7:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:2:0 8:7:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:2:0 8:7:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:2:0 8:7:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:2:0 8:8:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:2:0 8:8:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:2:0 8:8:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:2:0 8:8:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:2:0 9:0:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:2:0 9:0:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:2:0 9:0:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:2:0 9:0:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:2:0 9:1:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:2:0 9:1:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:2:0 9:1:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:2:0 9:1:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:2:0 9:2:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:2:0 9:2:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:2:0 9:2:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:2:0 9:2:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:2:0 9:3:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:2:0 9:3:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:2:0 9:3:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:2:0 9:3:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:2:0 9:4:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:2:0 9:4:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:2:0 9:4:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:2:0 9:4:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:2:0 9:5:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:2:0 9:5:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:2:0 9:5:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:2:0 9:5:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:2:0 9:6:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:2:0 9:6:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:2:0 9:6:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:2:0 9:6:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:2:0 9:7:0 " --planet-list } - There is a Class Association Rul
```

```
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:2:0 9:7:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:2:0 9:7:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:2:0 9:7:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:2:0 9:8:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:2:0 9:8:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:2:0 9:8:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:2:0 9:8:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5"  --location=" x 77:3:0 7:0:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-30 10:10:10 5"  --location=" x 77:3:0 7:0:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-1 10:10:10 5"  --location=" x 77:3:0 7:0:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-12-2 10:10:10 5"  --location=" x 77:3:0 7:0:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-11-29 10:10:10 5"  --location=" x 77:3:0 7:1:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-30 10:10:10 5"  --location=" x 77:3:0 7:1:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-1 10:10:10 5"  --location=" x 77:3:0 7:1:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-12-2 10:10:10 5"  --location=" x 77:3:0 7:1:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-11-29 10:10:10 5"  --location=" x 77:3:0 7:2:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-30 10:10:10 5"  --location=" x 77:3:0 7:2:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-1 10:10:10 5"  --location=" x 77:3:0 7:2:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-12-2 10:10:10 5"  --location=" x 77:3:0 7:2:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-11-29 10:10:10 5"  --location=" x 77:3:0 7:3:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-30 10:10:10 5"  --location=" x 77:3:0 7:3:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-1 10:10:10 5"  --location=" x 77:3:0 7:3:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-12-2 10:10:10 5"  --location=" x 77:3:0 7:3:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-11-29 10:10:10 5"  --location=" x 77:3:0 7:4:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-30 10:10:10 5"  --location=" x 77:3:0 7:4:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-1 10:10:10 5"  --location=" x 77:3:0 7:4:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-12-2 10:10:10 5"  --location=" x 77:3:0 7:4:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-11-29 10:10:10 5"  --location=" x 77:3:0 7:5:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-30 10:10:10 5"  --location=" x 77:3:0 7:5:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-1 10:10:10 5"  --location=" x 77:3:0 7:5:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-12-2 10:10:10 5"  --location=" x 77:3:0 7:5:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-11-29 10:10:10 5"  --location=" x 77:3:0 7:6:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-30 10:10:10 5"  --location=" x 77:3:0 7:6:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-1 10:10:10 5"  --location=" x 77:3:0 7:6:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-12-2 10:10:10 5"  --location=" x 77:3:0 7:6:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-11-29 10:10:10 5"  --location=" x 77:3:0 7:7:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-30 10:10:10 5"  --location=" x 77:3:0 7:7:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-1 10:10:10 5"  --location=" x 77:3:0 7:7:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-12-2 10:10:10 5"  --location=" x 77:3:0 7:7:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-11-29 10:10:10 5"  --location=" x 77:3:0 7:8:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-30 10:10:10 5"  --location=" x 77:3:0 7:8:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-1 10:10:10 5"  --location=" x 77:3:0 7:8:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-12-2 10:10:10 5"  --location=" x 77:3:0 7:8:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:3:0 8:0:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:3:0 8:0:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:3:0 8:0:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:3:0 8:0:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:3:0 8:1:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:3:0 8:1:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:3:0 8:1:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:3:0 8:1:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:3:0 8:2:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:3:0 8:2:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:3:0 8:2:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:3:0 8:2:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:3:0 8:3:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:3:0 8:3:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:3:0 8:3:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:3:0 8:3:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:3:0 8:4:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:3:0 8:4:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:3:0 8:4:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:3:0 8:4:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:3:0 8:5:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:3:0 8:5:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:3:0 8:5:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:3:0 8:5:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:3:0 8:6:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:3:0 8:6:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:3:0 8:6:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:3:0 8:6:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:3:0 8:7:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:3:0 8:7:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:3:0 8:7:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:3:0 8:7:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:3:0 8:8:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:3:0 8:8:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:3:0 8:8:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:3:0 8:8:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:3:0 9:0:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:3:0 9:0:0 " --planet-list } - There is a Class Association Rul
```

```
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:3:0 9:0:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:3:0 9:0:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:3:0 9:1:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:3:0 9:1:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:3:0 9:1:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:3:0 9:1:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:3:0 9:2:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:3:0 9:2:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:3:0 9:2:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:3:0 9:2:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:3:0 9:3:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:3:0 9:3:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:3:0 9:3:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:3:0 9:3:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:3:0 9:4:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:3:0 9:4:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:3:0 9:4:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:3:0 9:4:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:3:0 9:5:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:3:0 9:5:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:3:0 9:5:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:3:0 9:5:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:3:0 9:6:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:3:0 9:6:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:3:0 9:6:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:3:0 9:6:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:3:0 9:7:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:3:0 9:7:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:3:0 9:7:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:3:0 9:7:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5.5"  --location=" x 77:3:0 9:8:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-11-30 10:10:10 5.5"  --location=" x 77:3:0 9:8:0 " --planet-list } - There is a Class Association Rul
{  --date="2016-12-1 10:10:10 5.5"  --location=" x 77:3:0 9:8:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-2 10:10:10 5.5"  --location=" x 77:3:0 9:8:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-29 10:10:10 5"  --location=" x 77:4:0 7:0:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-30 10:10:10 5"  --location=" x 77:4:0 7:0:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-1 10:10:10 5"  --location=" x 77:4:0 7:0:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-12-2 10:10:10 5"  --location=" x 77:4:0 7:0:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-11-29 10:10:10 5"  --location=" x 77:4:0 7:1:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-30 10:10:10 5"  --location=" x 77:4:0 7:1:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-1 10:10:10 5"  --location=" x 77:4:0 7:1:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-12-2 10:10:10 5"  --location=" x 77:4:0 7:1:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-11-29 10:10:10 5"  --location=" x 77:4:0 7:2:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-11-30 10:10:10 5"  --location=" x 77:4:0 7:2:0 " --planet-list } - There is a Class Association Rule
{  --date="2016-12-1 10:10:10 5"  --location=" x 77:4:0 7:2:0 " --planet-list } - There is a Class Association Rule m
{  --date="2016-12-2 10:10:10 5"  --location=" x 77:4:0 7:2:0 " --planet-list } - There is a Class Association Rule m

-----------------------------------------------------------------------------------------------------------------------
314. (THEORY) Pseudorandomness, Voter Decision Error and lambda-tolerant Randomized Decision Trees of Voter Boolean F
-----------------------------------------------------------------------------------------------------------------------
Pseudorandom generators are those which stretch a seed l(n) to n : f:{0,1}^l(n)->{0,1}^n indistinguishable from perfe
with bounded error. References 314.1 and 314.2 describe a counting argument to show that there are PRGs. For a seed l

References:
-----------
314.1 Definition of Pseudorandomness - Simpler Distinguisher - http://people.seas.harvard.edu/~salil/pseudorandomness
314.2 Definition of Pseudorandomness - Counting argument - http://www.cse.iitk.ac.in/users/manindra/survey/complexity
314.3 Lambda-tolerant Randomized Boolean Decision Trees - https://www.math.u-szeged.hu/~hajnal/research/papers/dec_su

-----------------------------------------------------------------------------------------------------------------------
315. (FEATURE-DONE and BUG-STABILITY ISSUES) NeuronRain AsFer Commits for Virgo Linux - Commits - 25 August 2016
-----------------------------------------------------------------------------------------------------------------------
Kernel Panic investigation for boost::python c++ invocations of Virgo System Calls
-----------------------------------------------------------------------------------------------------------------------
1.Python code in AsFer that invokes Virgo Linux system calls by boost::python C++ bindings is being investigated furt
2.kern.log with 3 iterations of virgo_malloc()+virgo_set()+virgo_get() invocations succeeded by panics in random poin
3.Logs also contain the gdb vmlinux debugging showing line numbers within kernel source where panics occur. Prima fac

-----------------------------------------------------------------------------------------------------------------------
316. (FEATURE-DONE and BUG-STABILITY ISSUES) NeuronRain AsFer Commits for Virgo Linux - Commits - 26 August 2016
-----------------------------------------------------------------------------------------------------------------------
Kernel Panic investigation for boost::python c++ invocations of Virgo System Calls
-----------------------------------------------------------------------------------------------------------------------
kern.log with panic in FS code after boost::python C++ AsFer-VIRGO Linux system calls invocations

-----------------------------------------------------------------------------------------------------------------------
317. (THEORY) Generic Definitions of Majority and Non-majority Choice Errors and some contradictions - related to 14
-----------------------------------------------------------------------------------------------------------------------
Majority Voting Social Choice has usual meaning of aggregation of for and against votes. Non-majority Social Choice c
flavours:
       317.1 Pseudorandom Choice - a Pseudorandom Generator chooses a voting function from a set of Voting functions
```

```
5485        317.2 Social choice by ranking - set of boolean functions are ranked ascending in their error probabilities an
5486
5487    --------------------
5488    Algorithm for 317.1:
5489    --------------------
5490    There are pseudorandom generators in NC(Applebaum PRG, Parallel PRGs etc.,) which choose a boolean function at random
5491            - Invoke a PRG in NC or P to obtain pseudorandom bits X.
5492            - Choose an element in electorate indexed by X.
5493            - Goodness of LHS is equal to the goodness of chosen element.
5494
5495    LHS is BPNC algorithm to RHS BPEXP Condorcet Jury Theorem (CJT) unbounded circuit and if CJT converges to 1 in RHS an
5496
5497    --------------------
5498    Algorithm for 317.2:
5499    --------------------
5500            - foreach(voter)
5501            - {
5502            -       Interview the voter boolean function : Rank the voters by merit (error probabilities e.g noise stabili
5503            - }
5504            - choose the topmost as non-majority social choice
5505
5506    LHS is a PSPACE-complete algorithm and RHS is either BPEXP or EXP algorithm depending on convergence or divergence of
5507
5508    Above loop can be parallelized and yet it is in PSPACE. Proof of CJT circuit in RHS to be EXP-complete would immediat
5509             (*) Let there be exponential number of voters (i.e exponential in number of variables)
5510             (*) Number of voters = Number of steps in Bounded Halting Problem input Turing Machine = exponential
5511             (*) Voting halts when all exponential voters have exercised franchise applying their SAT = EXPTIME Turing Mac
5512
5513    If there are n boolean functions in total and probability that a boolean function i with goodness e(i) is chosen is:
5514            = x(e(i))/n, where x(e(i)) is the number of boolean functions with goodness e(i)
5515    When voter boolean functions have unequal error probabilities, the distribution is Poisson Binomial (which is for ber
5516
5517    Thus most generic Majority voting case is when:
5518            317.3 Voters have unequal decision errors
5519            317.4 Set of voter boolean functions is an assorted mix of varied error probabilities ranging from 0 to 1.
5520            317.5 Modelled by Poisson Binomial Distribution
5521
5522    In both Majority and Non-majority Choice, error of a boolean function comprises:
5523            317.6 Voting Error - Misrecorded votes, Probabilistic CNFs
5524            317.7 Voter Error - Noise Sensitivity of boolean function which has extraneous reasons like correlated flipped
5525    Randomized decision tree evaluation in 317.7 adds one additional scenario to error matrix of 8 possibilities in 14 an
5526
5527    As mentioned elsewhere in this document and disclaimer earlier, LHS circuit is a lowerbound to RHS circuit for a C-co
5528
5529    From https://www.math.ucdavis.edu/~greg/zoology/diagram.xml, following chain of inclusions are known:
5530            NC in BPNC in RNC in QNC in BQP in DQP in EXP
5531            RP in BPP in BQP in DQP in EXP
5532
5533    If LHS is a BPNC or RNC or BPP or RP pseudorandom algorithm to unbounded CJT RHS EXP-complete problem under equal goo
5534            EXP=BPNC (or) EXP=RNC (or) EXP=RP (or) EXP=BPP
5535    This is rather a serious collapse because the containments include Quantum classes BQP and QNC, implying classical pa
5536            Let number of voter decision functions with goodness xi = m(xi). Thus N = m(x1)+m(x2)+m(x3)+...+m(xn)
5537            Expected goodness of a PRG choice is:
5538                    1/N * summation(xi*m(xi)) = (x1*m(x1) + x2*m(x2) + x3*m(x3) + ... + xn*m(xn)) / N
5539            When all voter functions have goodness 1 then PRG choice in LHS of P(good) has goodness 1.
5540    For p=0.5, RHS goodness is 0.5 and LHS goodness is 0.5 => LHS is in BPNC or RNC or RP or BPP and RHS is BPEXP. If RHS
5541            EXP=BPNC (or) EXP=RNC (or) EXP=RP (or) EXP=BPP
5542    when goodness p=1 uniformly for all infinite homogeneous electorate.
5543
5544    References:
5545    ----------
5546    317.8 Derandomization Overview - [Kabanets] - http://www.cs.sfu.ca/~kabanets/papers/derand_survey.pdf
5547    317.9 Unconditional Lowerbounds against advice - [BuhrmanFortnowSanthanam] - http://homepages.inf.ed.ac.uk/rsanthan/h
5548    317.10 EXPTIME-Completeness - [DingZhuDu-KerIKo] - Theory of Computational Complexity - https://books.google.co.in/bo
5549    317.11 BPP with advice - http://people.cs.uchicago.edu/~fortnow/papers/advice.pdf - "...It can be shown using a trans
5550
5551    ------------------------------------------------------------------------------------------------------------------------
5552    318. (THEORY) Yao's XOR Lemma, Hardness of Pr(Good) Majority Voting Function Circuit and Majority Version of XOR Lemm
5553    ------------------------------------------------------------------------------------------------------------------------
5554    Caution: Majority Hardness Lemma derivation is still experimental with possible errors.
5555
5556    Hardness of a boolean function is defined as how hard computationally it is to compute the function with a circuit of
5557             if Pr(C(x) != f(x)) = delta where C(x) computes boolean function f(x), then f(x) delta-hard.
5558    Majority function is known to have formula of size O(n^5.3) - from Sorting networks of [AjtaiKomlosSzemeredi] and non
5559
5560    Hardness Amplification result by [AndrewYao] states that a strongly-hard boolean function can be constructed from mil
5561             318.1 Let f be a boolean function with delta-hardness. Therefore there is a circuit C(x) with size s,such tha
5562             318.2 Each f(xi) in XOR function f1 in 318.1, is either flipped or not flipped. Let Zi be the random variable
5563             318.3 Pr(Good) majority voting boolean function computes composition of Majority function with Voting functio
5564             318.4 For simplicity, it is assumed that all voters have same boolean function with hardness delta computable
5565             318.5 Majority function makes error when the inputs are flipped by some correlation. This Probability that Pr
```

```
        318.6 Noise Sensitivity of Majority function is O(1/sqrt(n*epsilon)) where epsilon is probability of flip per
        318.7 Probability that Majority function is computed incorrectly = NoiseSensitivity +/- randomerror = [c/sqrt
        318.8 Above derivation,if error-free,is the most important conclusion derivable for hardness of Pr(Good) majo
        318.9 Therefore from 318.7, Pr(C(Majn(f(x1),f(x2),...,f(n)) != Majn(f(x1),f(x2),...,f(n))] = [c/sqrt(n*delta)
        318.10 Hardness is amplified if [c/sqrt(n*delta) +/- randomerror] >= delta as follows:

                Hardness of Maj+voter composition       [c/sqrt(n*delta)] +/- randomerror
                --------------------------------  =  ----------------------------  >>  1
                Hardness of voter function                 delta

        318.11 Let randomerror=r. From above, [c/sqrt(n*delta) +/- r] / delta must be >> 1 for hardness amplification

     Hardness of Maj+voter composition       [c/sqrt(n*delta)] + [sum(column2 error entries)] - [sum(column3 no error
    ---------------------------------  =  ------------------------------------------------------------------------
         Hardness of voter function                                        delta

-----------------------------------------------------------------------------------------------------------------
For large n, c/sqrt(n*delta) tends to 0 and If [sum(column2 error entries) -  sum(column3 no error entries)] >> delta
-----------------------------------------------------------------------------------------------------------------
        318.12 318.11 implies that for weakly hard functions (low delta), if quantity in numerator (hardness of major
        318.13 318.11 also implies that Majority+Voter composition is extremely hard to compute concurring with expon
        318.14 Being extremely hard implies that RHS of Pr(Good) could be one-way function. One-way functions are har
                Pr(f(f_inverse(y))=f(x)) is almost 0.
For example inverse for Pr(Good) majority voting function is the one that returns set of all voters who voted for a c
        318.15 Conjecture: Circuit for reverse direction (i.e decomposition of majority to voters who voted in favour
        318.16 Huge hardness of Pr(Good) RHS also implies that very strong pseudorandom number generators can be cons
        318.17 The counting problem in 318.15 finds all possible permutations of voters who could have voted in favou
        318.18 318.17 implies Pr(Majority(Majorityinverse(1))==Majority(v1,v2,...,vn)) = 1/#SAT <= 1/2^n in worst cas
        318.19 In average case, expected number of SAT assignments to MajoritySAT = 1*1/2^n + 2*1/2^n + ...+ 2^n*1/2^
        318.20 318.19 implies Majority is one-way. Therefore FP != FNP and thus P != NP if hardness is amplified in 3

References:
-----------
318.20 [Impagliazzo-Wigderson] - P=BPP if EXP has 2^Omega(n) size circuits - http://www.math.ias.edu/~avi/PUBLICATIOM
318.21 [Trevisan] - XOR Lemma Course Notes - http://theory.stanford.edu/~trevisan/cs278-02/notes/lecture12.ps
318.22 [Goldreich-RaniIzsak] - One-way functions and PRGs - http://www.wisdom.weizmann.ac.il/~oded/PDF/mono-cry.pdf
318.23 Noise Sensitivity of Majority Function - http://pages.cs.wisc.edu/~dieter/Courses/2008s-CS880/Scribes/lecturei
318.24 Definition of One-way functions - https://en.wikipedia.org/wiki/One-way_function

-----------------------------------------------------------------------------------------------------------------
319. (THEORY) Integer Partitions, Multiway Contests, Hash Table Functions and Ordered Bell Numbers - related to 256 a
-----------------------------------------------------------------------------------------------------------------
Number of hash table functions derived in https://sites.google.com/site/kuja27/IntegerPartitionAndHashFunctions_2014.
all possible permutations of hash buckets per partition does not consider order of elements with in each bucket. Bell
Eric Temple Bell) is the number of all possible partitions of a set (induced by all possible equivalence relations).
table is a set of equivalence classes (i.e each bucket is an equivalence class and hash function is the relation), pa
Ordered Bell Number takes into account order of keys in each bucket and gives all possible Ordered Hash Table Functio
estimate of number of hash table functions than plain counting from integer partitions which is unordered. Unordered
given by: sigma_k=0_to_n({n,k}) where {n,k} is the Stirling Number of Second kind computing number of ways of partiti
into subsets of size exactly k.

Let there be n keys and size of hash table be m. Let hi(x) mod m be a hash function.
For each hi
{
        For each x < n
        {
                Compute hi(x) mod m and append to corresponding chain of buckets in hash table
        }
}

Above loops create all possible partitions of set of size n. If just the integer partition is reduced to above hash t
        sum_k=1_to_m(mP({n,k})) >= Ordered Bell Number = sum_k=1_to_m({n,k})

In multipartisan voting with more than 2 candidates, above augmented stirling number is also the number of all possib

Probability that above multiway majority function has an error (i.e chooses wrong outcome) =
Probability of changing one voting pattern to the other that changes ranking of candidates

Each voting pattern can be deemed to be a point vertex on an m-dimensional metric space and edges between these verti
patterns. If each edge has a probability of occurrence, Probability of flip in voting pattern due to a malpractice is

In a sense the multiway majority hashing previously is a locality sensitive hashing wherein similarity between items

As mentioned in previous paragraphs, each voting pattern is a point vertex on a metric space and it is necessary to c

Probability of multiway majority choosing a wrong outcome = Probability that voting pattern noise flip changes sorted

319.8 is an NVIDIA CUDA parallel implementation of hash table chaining. 319.9 is a recent parallel locality sensitive


References:
```

```
5647 ----------
5648 319.1 Unordered Bell Number - https://en.wikipedia.org/wiki/Bell_number
5649 319.2 Ordered Bell Number - https://en.wikipedia.org/wiki/Ordered_Bell_number
5650 319.3 Mining Massive Data Sets - [UllmanRajaramanLeskovec] - http://infolab.stanford.edu/~ullman/mmds/book.pdf
5651 319.4 Lowerbounds for Locality Sensitive Hashing - [RyanODonnell] - https://www.cs.cmu.edu/~odonnell/papers/lsh.pdf
5652 319.5 Theory of Partitions - Bell numbers - [GeorgeEAndrews] - http://plouffe.fr/simon/math/Andrews%20G.E.%20The%20Th
5653 319.6 Power of Simple Tabulation Hashing - [PatrascuThorup] - http://arxiv.org/pdf/1011.5200v2.pdf - Lemma 4 - Balls
5654 319.7 Balls and Bins, Chained Hashing, Randomized Load balancing - http://pages.cs.wisc.edu/~shuchi/courses/787-F09/s
5655 319.8 Parallel Hash tables - Chaining buckets with arrays - http://idav.ucdavis.edu/~dfalcant//downloads/dissertatior
5656 319.9 Parallel Locality Sensitive Hashing - [Narayanan Sundaram???,Aizana Turmukhametova,Nadathur Satish???,Todd Most
5657
5658
5659 ------------------------------------------------------------------------------------------------------------------------
5660 320. (BUG-STABILITY ISSUES) Commits - 6 September 2016
5661 ------------------------------------------------------------------------------------------------------------------------
5662 VIRGO Linux Kernel Stability Analysis - 2 September 2016 and 6 September 2016
5663 ------------------------------------------------------------------------------------------------------------------------
5664 kern.log(s) for Boost::Python invocation of VIRGO system calls with and without crash on two dates have been committe
5665 Logs on 2/9/2016 have a crash as usual in VM paging scatterlist.c.Today's invocation logs show perfect execution of w
5666
5667 ------------------------------------------------------------------------------------------------------------------------
5668 321. (BUG-STABILITY ISSUES) Commits - 8 September 2016
5669 ------------------------------------------------------------------------------------------------------------------------
5670 Continued kernel panic analysis of boost::python-VIRGO system calls invocations. Similar pattern of crashes in vma pa
5671 successful panic-free invocation in the end.
5672
5673 ------------------------------------------------------------------------------------------------------------------------
5674 322. (BUG-STABILITY ISSUES) Commits - 9 September 2016
5675 ------------------------------------------------------------------------------------------------------------------------
5676 Boost::python-VIRGO system calls invocation kernel panic analysis showing some problem with i915 graphics driver inte
5677 VIRGO system calls.
5678
5679 ------------------------------------------------------------------------------------------------------------------------
5680 323. (THEORY) Complement Functions for Hash Chains, PAC Learning, Chinese Remaindering and Post Correspondence Proble
5681 - important draft additions to http://arxiv.org/pdf/1106.4102v1.pdf
5682 ------------------------------------------------------------------------------------------------------------------------
5683 A hash table chain can be viewed as a subset segment of a larger rectangular region. Complement of this subset is anc
5684 For example following schematic ilustrates hash table complements with respect to chained buckets:
5685
5686         ------------------------ ####################
5687         ---------- ##################################
5688         ---------------- ############################
5689         ------------------ ##########################
5690         ------ ######################################
5691
5692 Above diagram has two hash table chains shown in different constituent colors. Together these two cover a rectangular
5693         f(x01) mod m = 0
5694         f(x11) mod m = 1
5695         f(x12) mod m = 1
5696         ...
5697         f(x21) mod m = 2
5698         f(x22) mod m = 2
5699         ...
5700 where xij is the j-th element in i-th chain part in f2 on right. Above tabulation can be solved by Euclid's algorithm
5701         [-m*y + f(x11)] = 1 => f(x11) mod m = 1
5702 Therefore above table can be written as system of congruences:
5703         f(x01) = a01*m + 0
5704         f(x11) = a11*m + 1
5705         f(x12) = a12*m + 1
5706         ...
5707         f(x21) = a21*m + 2
5708         f(x22) = a22*m + 2
5709         ...
5710 Since m is known and all aij can take arbitrary values, complement function for f2 can be constructed by fixing aij a
5711
5712 Boolean 0-1 majority function is a special case of multiway majority function (or) LSH, where hashtable is of size 2
5713         Augmented Stirling Number of Boolean Majority = 2P(n,1} + 2P{n,2}
5714
5715 Chinese Remaindering Theorem states that there is a ring isomorphism for N=n1*n2*n3*....*nk (for all coprime ni) such
5716         X mod N <=> (X mod n1, X mod n2, X mod n3, ...., X mod nk)
5717         (or)
5718         Z/NZ <=> Z/n1Z * Z/n2Z * .... * Z/nkZ
5719 Chinese Remaindering has direct application in hash table chains as they are created in modular arithmetic. K hash fu
5720
5721 From Post Correspondence Problem, if f1 and f2 in schematic diagram are two voting patterns, finding a sequence such
5722         p1p2p3...pm = q1q2q3....qm
5723 is undecidable where pi and qi are parts in voting patterns f1 and f2 as concatenated strings. What this means is tha
5724 which makes serialized voters in both patterns equal is undecidable.
5725
5726 Alternative proof of undecidability of Complement Function Construction with PCP :
5727 ------------------------------------------------------------------------------
```

```
Complement Functions are reducible to Post Correspondence Problem. PCP states that finding sequence of numbers i1,i2,
        s(i1)s(i2)s(i3)....s(ik) = t(i1)t(i2)t(i3)...t(ik)
where s(ik) and t(ik) are strings is undecidable.

Example inductive base case:
----------------------------
Let f(x) = 2,4,6,8,....
and g(x) = 1,3,5,7,....
f(x) and g(x) are complements of each other.

Define a(i) and b(i) as concatenated ordered pairs of f(xi) and g(xi):
        a1 = 2          b1 = 2,3
        a2 = 3,4        b2 = 4,5
        a3 = 5,6        b3 = 6,7
        a4 = 7,8        b4 = 8,9
        a5 = 9,10       b5 = 10
[e.g a2 = 3, 4 where g(a2) = 3 and f(a2) = 4; b3= 6,7 where f(b3) = 6 and g(b3) = 7 and so on. Here complement of an

Then, 1,2,3,4,5 is a sequence such that:
        a1a2a3a4a5 = b1b2b3b4b5 = 2,3,4,5,6,7,8,9,10
which can be parenthesised in two ways as:
        2(3,4)(5,6)(7,8)(9,10) = (2,3)(4,5)(6,7)(8,9)10 = 2,3,4,5,6,7,8,9,10
Each parenthesization is a string representation of a possible way to construct a complement function. Ideally, proce
        f(x) = 2,3,5,7,11,...
        g(x) = 4,6,8,9,10,12,...

        ([2,3],4)(5,6)(7,[8,9,10])(11) = ([2,3])(4,5)(6,7)([8,9,10],11) = 2,3,4,5,6,7,8,9,10,11

Generic inductive case:
-----------------------
        (f(x1),succ(f(x1))(f(x2),succ(f(x2))).... = (predec(g(x1),g(x1))(predec(g(x2),g(x2)).... = Z
        (or)
        concatenation_of(f(xi),g(xi)) = concatenation_of(g(xi),f(xi)) = Z
where succ() and predec() are successor and predecessor functions respectively in lambda calculus jargon and xi take

Two complementation parenthesizations can be drawn as step function where indentation denotes complementary g points

                        |
                        | f(1)
                        |
                        ---------
                                |
                                | g(1)
                                |
                        ---------
                        |
                        | f(0)
                        |
                        ---------
                                |
                                | g(0)
                                |
                                |
                        ---------
                        |
                        | f(2)
                        ---------
                                |
                                | g(2)
                                |

which represent two parenthesization correspondences (sequence on right is staggered by one segment):
       (f(1),g(1))(f(0),g(0))(f(2),g(2)) = (f(1))(g(1),f(0))(g(0),f(2))(g(2)) = f(1)g(1)f(0)g(0)f(2)g(2)
Hence x1,x2,x3 take values 1,0,2 for string on left. String on the right despite staggering also takes values 1,0,2 f

Two Generic parenthesizations for f and its complement g are:
       [f(x1)][g(y1)f(x2)][g(y2)f(x3)][g(y3)]... = [f(x1)g(y1)][f(x2)g(y2)][f(x3)g(y3)]... = f(x1)g(y1)f(x2)g(y2)f(x3
where each f(xi) and g(yi) are contiguous streak of set of values of f and g. xi is not necessarily equal to yi. Requ
x1-y1, x2-y2, x3-y3, ... in right and x1, y1-x2, y2-x3, ... in left to make the strings correspond to each other. Her

Another alternative definition of complement function is: function f and its complement g are generating functions fo

Complementation Disjoint Set Cover can be represented as bipartite graph where edges are between two sets A and B. Se

Boolean 0-1 majority special case of Hash chains and Complement Functions:
---------------------------------------------------------------------------
A hash table of size 2 with 2 chains partitions the set of keys into 2 disjoint sets. Each chain in this hash table i

-------------------------------------------------------------------------------------------------------------------
324. (FEATURE-DONE) PAC Learning for Prime Numbers encoded as binary strings - Commits - 12 September 2016
-------------------------------------------------------------------------------------------------------------------
```

PAC Learning implementation has been augmented to learn patterns in Prime Numbers encoded as binary strings. For each

---

325. (FEATURE-DONE) PAC Learning for Prime Numbers - Commits - 14 September 2016

Some errors corrected in bit positions computation in JSON mappings for PAC learning of Prime Numbers. Logs committed

---

326. (BUG-STABILITY ISSUES) Boost::Python-VIRGO System calls invocations random kernel panics - Commits - 15 Septembe

---

Further Kernel Panic analysis in i915 driver for Boost::python-VIRGO system calls invocations.

---

327. (THEORY) Algorithmic Fairness, Pr(Good) Majority Voting Circuit and Algorithmic Decision Making - Related to 14,

---

In Majority Voting hardness analyzed thus far, voting functions of Individual Human Voters are assumed. Recent advanc
decision making (High Frequency Algorithmic Trading, Predictive Policing etc.,) involve decision making by algorithms
implicit so far that algorithms cannot have bias. But there is a new emerging field of algorithmic fairness which hig
bias by machine learning algorithms in decision making (bias could be in training dataset, algorithm's false assumpti
wrong conclusions based on correlations etc.,) that could subvert stock trading buy-sell decisions, criminal justice
 unfairness could happen in Majority voting also if the voters are algorithms (algorithm internally using a boolean f
function, past training data to make future decisions among others). Unfair voting algorithms imply that Pr(Good) sum
converge to 100% and therefore error is non-zero. Unfair voting is detrimental to distributed cloud computing involvi
choice - e.g loadbalancing of requests get skewed to a node unfairly by bad voting. Most importantly a proof of exis
voting algorithm implies that LHS of Pr(Good) summation is 1 (From 317.2 least error algorithm is chosen as Non-major

References:

327.1 Algorithmic Fairness - https://*algorithmicfairness.wordpress.com*
327.2 Leader Election Algorithms in Cloud - HBase ZooKeeper - [Mahadev Konar - Yahoo] - http://*wiki.apache.org/hadoop*
327.3 Leader Election and Quorum in ElasticSearch - https://*www.elastic.co/blog/found-leader-election-in-general* - *Ea*

---

328. (FEATURE-DONE) Locality Sensitive Hashing Implementation - Nearest Neighbours Search - Commits - 16 September 20

1.This commit implements locality sensitive hashing in python by wrapping defaultdict with hashing and distance measu
2.Locality Sensitive Hashing is useful for clustering similar strings or text documents into same bucket and is thus
classifier and an inverted index too.
3.In this implementation, very basic LSH is done by having replicated hashtables and hashing a document to each of th
a random polynomial hash function which itself is aggregation of random monomials.
4.For a query string, random hash function is again computed and buckets from all hashtables corresponding to this ha
nearest neighbour set.
5.Each of these buckets are sieved to find the closest neighbour for that hashtable.
6.Sorting the nearest neighbours for all hash tables yields a ranking of documents which are in the vicinity of the c
7.The input strings are read from a text file LocalitySensitiveHashing.txt
8.Logs with hashtable dumps and nearest neighbour rankings are in testlogs/LocalitySensitiveHashing.out.16September20

---

329. (FEATURE-DONE) LSH WebCrawler Support - Commits - 19 September 2016

Locality Sensitive Hashing now accepts scrapy crawled webpages as datasources.

---

330. (BUG - STABILITY ISSUES) Boost::Python AsFer-VIRGO system calls kernel panics - Commits - 19 September 2016

Boost::python AsFer - VIRGO system call ongoing kernel panic analysis - i915 DRM race condition kernel panic in VM pa

---

331. (FEATURE-DONE) ZeroMQ based Concurrent Request Servicing CLI - Client and Multithreaded Server Implementation

---

Commits - 21 September 2016

1.NeuronRain already has support for RESTful GUI implemented in Python Tornado and NeuronRain code can be executed by
filling up **HTML** form pages.
2.ZeroMQ has a lowlevel highly performant low latency concurrency framework for servicing heavily concurrent requests
3.ZeroMQ is a wrapper socket implementation with special support for Request-Reply, Router-Worker, Pub-Sub design pat
4.Important advantage of ZeroMQ is lack of necessity of lock synchronization (i.e ZeroMQ is lock-free per its documen
for consistency of concurrent transactions
5.Hence as a CLI alternative to HTTP/REST GUI interface,a C++ client and server have been implemented based on ZeroMQ
Router-Dealer-Worker sockets pattern to serve concurrent requests.
6.ZeroMQ client: ./zeromq_client "<neuronrain executable command>"
7.ZeroMQ server: invokes system() on the executable arg from zeromq client
8.With this NeuronRain has following interfaces:
        8.1 telnet client --> NeuronRain VIRGO ports
        8.2 VIRGO system call clients --> NeuronRain VIRGO ports
        8.3 AsFer boost::python VIRGO system call invocations ----------------> NeuronRain VIRGO ports
        8.4 Tornado GUI RESTful --> NeuronRain VIRGO ports
        8.5 ZeroMQ CLI client/server ---> NeuronRain VIRGO ports

---

```
5890  332. (FEATURE-DONE) KingCobra VIRGO Linux workqueue and Kafka Publish-Subscribe Backend Message Queue support in Stre
5891  ----------------------------------------------------------------------------------------------------------------------
5892  Commits - 22 September 2016
5893  ----------------------------------------------------------------------------------------------------------------------
5894  1.Streaming_AbstractGenerator has been updated to include KingCobra request-reply queue disk persisted store, as
5895  a data storage option (reads /var/log/kingcobra/REQUEST_REPLY.queue). Initial design option was to integrate a Kafka
5896  KingCobra kernelspace servicerequest() function which upon receipt of a message from linux kernel workqueue,also publ
5897  to a Kafka Topic. But tight coupling of Kafka C Client into KingCobra is infeasible because of conflicts between user
5898  files of Kafka and kernelspace include header files of Linux. This results in compilation errors. Hence it looks suff
5899  persisted KingCobra REQUEST_REPLY.queue by a standalone Kafka python client and publish to Kafka subscribers. This le
5900  by python code on KingCobra queue.
5901  2.NeuronRain backend now has a support for Kafka Pub-Sub Messaging.
5902  3.New publisher and subscriber for Kafka (with Python Confluent Kafka) have been written to read data from
5903  Streaming_AbstractGenerator and to publish/subscribe to/from a Kafka Message Broker-Topic. Thus any datasource that
5904  AsFer may have(file,HBase,Cassandra,Hive etc.,) is abstracted and published to Kafka. This also unites AsFer backend
5905  and KingCobra disk persistence into a single Kafka storage.
5906
5907  ----------------------------------------------------------------------------------------------------------------------
5908  333. (BUG-STABILITY ISSUES) Boost::Python AsFer-VIRGO system call invocations kernel panic ongoing investigation
5909  ----------------------------------------------------------------------------------------------------------------------
5910  Commits - 23 September 2016
5911  ----------------------------------------------------------------------------------------------------------------------
5912  Ongoing Boost::Python AsFer-VIRGO system call invocation kernel panic analysis:
5913  Random crashes remain, but this time no crash logs are printed in kern.log and finally a successful invocation happen
5914  same as i915 GEM DRM crash similar to earlier analyses.
5915
5916  Pattern observed is as follows:
5917          1. First few invocations fail with virgo_get() though virgo_malloc() and virgo_set() succeed.
5918          2. After few failures all virgo calls succeed - virgo_malloc(), virgo_set() and virgo_get() work without any
5919          3. Sometimes virgo_parse_integer() logs and few other logs are missing. When all logs are printed, success ra
5920          4. Some failing invocations have NULL parsed addresses. When there are no NULL address parsings, success rate
5921          5. a rare coincidence was observed: Without internet connectivity crashes are very less frequent. (Intel Micr
5922  updates playing spoilsport again?).
5923          6. There have been panic bugs reported on i915 GEM DRM and recent patches for busy VMA handling to it:
5924                  6.1 https://bugs.launchpad.net/ubuntu/+source/linux/+bug/1492632
5925                  6.2 https://lists.freedesktop.org/archives/intel-gfx/2016-August/102160.html
5926          7. Intel GPU i915 GEM DRM docs - https://01.org/linuxgraphics/gfx-docs/drm/gpu/drm-mm.html
5927
5928  ----------------------------------------------------------------------------------------------------------------------
5929  334. (BUG-STABILITY ISSUES) Boost::Python AsFer-VIRGO systemcall/drivers invocations kernel panics - new findings
5930  ----------------------------------------------------------------------------------------------------------------------
5931  Commits - 28 September 2016
5932  ----------------------------------------------------------------------------------------------------------------------
5933  Continued analysis of kernel panics after VIRGO systemcalls/drivers code in i915 GPU driver. Has hithertoo unseen str
5934  memory details have been committed to cpp_boost_python_extensions/testlogs/.
5935
5936  ----------------------------------------------------------------------------------------------------------------------
5937  335. (BUG-STABILITY ISSUES) Boost::Python AsFer-VIRGO systemcall invocations panics - more findings
5938  ----------------------------------------------------------------------------------------------------------------------
5939  Commits - 29 September 2016
5940  ----------------------------------------------------------------------------------------------------------------------
5941  Kernel Panic Analysis for Boost::Python AsFer - VIRGO system calls invocations:
5942  This log contains hitherto unseen crash in python itself deep within kernel (insufficient logs) followed by random -3
5943  errors. Finally successful invocations happened. Connections between -32,-107 errors and random panics/freezes were a
5944  ago (Blocking and Non-blocking socket modes). i915 DRM related stacks were not found in kern.log. Random disappearanc
5945  logs is quite a big travail. Crash within python could be i915 related - cannot be confirmed without logs. Pattern em
5946  of: Something wrong going on between CPU and GPU while allocating kernel memory in CPU domain - kmalloc() is likely a
5947  GPU and not CPU - quite a weird bug and unheard of.
5948
5949  ----------------------------------------------------------------------------------------------------------------------
5950  336. (BUG-STABILITY ISSUES) VIRGO kernel panics - final findings - 30 September 2016
5951  ----------------------------------------------------------------------------------------------------------------------
5952  Further kernel panic investigation in VIRGO - probably the last. Logs with analysis have been committed to cpp_boost_
5953  testlogs/.
5954
5955  ----------------------------------------------------------------------------------------------------------------------
5956  337. (THEORY) Ramsey Theorem, Edge Labelling of Voting Graph and Multiway Majority Function - 5 October 2016 - relate
5957  ----------------------------------------------------------------------------------------------------------------------
5958  Multiway majority voting can be drawn as a directed graph. There is an edge between vertices v1 and v2 if v1 votes fo
5959  Voting Graph with a weight >= 0.  Realworld example of this is web link graph where incoming links to a webpage are v
5960  outgoing links from a webpage are votes for adjacent pages. PageRank is a special case of Multiway Majority Function
5961  candidate webpages by a converging random walk markov chain of transition probabilities, with a rider that all webpag
5962  and candidates making it a peer-to-peer majority voting. PageRank is thus a Non-boolean voting function. If web link
5963  of depth d, it is equivalent to depth-d recursive majority function.  Edge labelling of a graph assigns colors to edg
5964  (Edge coloring is a special case of labelling with restriction no two adjacent edges are of same color). Voting graph
5965  labelled where each color of an edge denotes a voter affiliation. Ramsey Number in combinatorics states that there ex
5966  v=RN(r,s) for every graph of order v, there exists a clique of size r or independent set of size s. For any Voting gr
5967  implies emergence of a clique or an independent set i.e voters who vote among themselves or who do not vote for each
5968  graph is complete then Ramsey Theorem implies emergence of monochromatic cliques r (red cliques) or s (blue cliques).
5969
5970  References:
```

```
------------
337.1 Ramsey Theorem Lecture Notes - http://math.mit.edu/~fox/MAT307-lecture05.pdf
337.2 Ramsey Theorem - https://en.wikipedia.org/wiki/Ramsey%27s_theorem

-----------------------------------------------------------------------------------------------------------------------
338. (THEORY) Van Der Waerden Number, Schur, Szemeredi and Ramsey Theorems, Coloring of Integers and Complement Funct
-----------------------------------------------------------------------------------------------------------------------
Complement Function over Integer sequences can be defined in terms of 2-colorings of the integers. Described previous
undecidability proof of complementation, a function f and its complement g can be construed as 2-colorings of the Dis
the set of natural numbers - each color is a function - for example f is red and g is blue.

Schur's Theorem for Ramsey coloring of integer sequences states (quoted from http://math.mit.edu/~fox/MAT307-lecture6

" ...  Schur's theorem
Ramsey theory for integers is about finding monochromatic subsets with a certain arithmetic structure. It starts with

Theorem 3.
For any k >= 2 , there is n > 3 such that for any k-coloring of 1,2,3,...,n, there are three integers x, y, z of the
x + y = z ...  "

For complement function special case (i.e 2-coloring of the sequences), Schur Theorem implies that there are always i

" ...  for any given positive integers r and k, there is some number N such that if the integers {1, 2, ..., N} are c

Van Der Waerden Theorem for 2-colorings of natural numbers is equivalent to complementation disjoint set cover of nat
arity). This implies arbitrarily long monochromatic arithmetic progressions can be found in prime-composite complemen

Finding the arithmetic progressions for Van Der Waerden numbers have been formulated as SAT instances which involves

Concept of complement graphs have been studied in Perfect Graph Theorem and has strong resemblance to function comple

A contrived independent set can be created by adding edges from constituent vertices of the maximum clique which is a

Any 2 graphs corresponding to 2 functions defined on same set obtained from previous construction are isomorphic. For

In the context of complement function graphs constructed previously, functions are mapped to maximum clique subgraph

Szemeredi's Theorem which is the generalization of Van Der Waerden's Theorem states: For any subset A of N of natural

Roth Estimate for 2-coloring of integer sequences (in references 338.4 and 338.5) is a measure of order or randomness
       |{f(x)}| = {N+N^(1/4+epsilon)} / 2
       |{g(x)}| = {N-N^(1/4+epsilon)} / 2
       with probability equal to natural density of arithmetic progression in minimax()
Previous approximation with sampling relates size of complement functions,coloring and arithmetic progressions by Rot

Complement Graphs F and G constructed previously for function f and its complement g have set of all values of invers

Another important question is: Can a complement function polynomial be constructed from constituent monochromatic ari
Let f1,f2,f3,... be functions and g1,g2,g3,... be their respective complements. They are represented as graphs constr
There are 2 possibilities of arithmetic progressions - APs have both colors (elements from both function and its comp
       AP1 = 2x+1 => 1,3,5,7,9,.....
       AP2 = 3x+1 => 1,4,7,11,14,....
Merging them gives 1,3,4,5,7,9,10,11,13.... which is a subset of 1,2,3,4,5,....
----------------------------
Scenario1: Monochromatic APs
----------------------------
Approximation polynomial for sequence interpolated from constituent monochromatic APs mergesorted = C(x). Let AP1(x),
----------------------------
Scenario2: Multichromatic APs
----------------------------
When APs have elements of both colors (from both functions fk() and gk()), parallel mergesort of APs and interpolatic

References:
----------
338.1 Bound for Van Der Waerden Numbers - https://www.emis.de/journals/INTEGERS/papers/a20int2005/a20int2005.pdf
338.2 Exact Ramsey Theory and Van Der Waerden Numbers by SAT-solvers - [OliverKullmann] - https://arxiv.org/pdf/1004.
338.3 Erdos-Turan Conjecture - https://en.wikipedia.org/wiki/Erd%C5%91s_conjecture_on_arithmetic_progressions - If su
elements of a set of positive integers diverges, then the set contains arbitrarily long arithmetic progressions
338.4 Roth Estimate - http://matwbn.icm.edu.pl/ksiazki/aa/aa9/aa9125.pdf - Consider a two coloring of set of natural
338.5 Roth Estimate is nearly sharp - Beck, J. Combinatorica (1981) 1: 319. doi:10.1007/BF02579452 - http://link.spri
338.6 Discrepancy in Arithmetic Progressions - [MatousekSpencer] - Roth estimate is best possible - ROTH(N) <= N^1/4
338.7 Discrepancy Minimization by Walking on Edges - [BansalLovettMeka] - https://arxiv.org/pdf/1203.5747v2.pdf - Thi
338.8 Fourier Interpolation of n data points with polynomial of degree m - http://www.muskingum.edu/~rdaquila/m350/fo
338.9 Fourier Analysis and Szemeredi's Theorem - Roth's Argument - Documenta Mathematica ICM Extra Volume 1998 - http
338.10 Hales-Jewett Theorem - [MichelleLee] - http://www.math.uchicago.edu/~may/VIGRE/VIGRE2009/REUPapers/Lee.pdf - g
338.11 Polynomial Interpolation from Low Discrepancy Arithmetic Progressions - [The Discrepancy Method: Randomness an
338.12 Sequences Generated By Polynomials - [CorneliusSchultz] - https://www.researchgate.net/profile/E_Cornelius_Jr/
338.13 Polynomial Points, Green-Tao Theorem and Arithmetic Progressions - [CorneliusSchultz] - https://cs.uwaterloo.c
338.14 Recent Advanced Theorems in k-coloring of integer sequences - http://people.math.sc.edu/lu/talks/ap4.pdf - the

-----------------------------------------------------------------------------------------------------------------------
```

339. (FEATURE-DONE) Boyer-Moore Streaming Majority Algorithm - Commits - 7 October 2016

This commit implements Boyer-Moore algorithm for finding Majority element in Streaming Sequences. It uses the Streami
Abstraction for input streaming datasource. Logs have been committed to python-src/testlogs/

340. (FEATURE-DONE) GSpan Graph Substructure Mining Algorithm Implementation - Commits - 13 October 2016 and 14 Octob

1.Graph Substructure Mining GSpan algorithm implementation with logs in testlogs/
2.This code requires the Graph vertices to be labelled by unique integers
3.Integer labelling of vertices makes it easier for DFSCode hashes to be generated uniquely.

341. (THEORY) Graph Mining Algorithms and Recursive Gloss Overlap Graph for text documents - Document Similarity

Recursive Gloss Overlap Algorithm implementations in python-src/InterviewAlgorithm and its Spark Cloud Variants gener
with word labels from Text Documents. GSpan algorithm implemented in (340) mines subgraphs and edges common across gr
This allows extraction of common patterns amongst text document graphs and thus is an unsupervised similarity cluster
text analytics. GSpan has provisions for assigning minimum support for filtering patterns which amounts to finding pr
keywords in recursive gloss overlap graphs.

342. (FEATURE-DONE) Graph Mining Recursive Gloss Overlap Graph for text documents - Document Similarity

Commits - 17 October 2016

1.Code changes have been done to choose between numeric and word labelling of Graph vertices in GSpan GraphMining imp
2.New file GraphMining_RecursiveGlossOverlap.py has been added to JSON dump the Recursive Gloss Overlap graph of a te
3.New directory InterviewAlgorithm/graphmining has been created which contains the numeric and word labelled Recusive
graphs of 5 example text documents in topic class "Chennai Metropolitan Area Expansion"
4.logs for commong edges mined between two Recursive Gloss Overlap document graphs has been committed in testlogs/
5.Spidered web text has been updated
6.GraphMining_RecursiveGlossOverlap.py JSON dumps the RGO edges into a text file and GraphMining_GSpan.py JSON loads
InterviewAlgorithm/graphmining/

343. (THEORY) Streaming Majority and 2-Coloring(Complement Functions) - related to 14.16 and 19,24,323,338,339 - impo
updates to http://arxiv.org/pdf/1106.4102.pdf - 1 November 2016

In simple 2 candidate majority voting done on stream of votes, a 2-colored sequence of votes by voters is generated.
already mentioned in infinite majority (Erdos Discrepancy Theorem) - 14.16.Each color symbolizes a candidate voted. T
the sequence of votes into 2 monochromatic sets. Obviously, if the voters are uniquely identified by a sequence numbe
theorems imply there are monochromatic and multichromatic arithmetic progressions in voter unique identities. Boyer-M
computes streaming majority (implemented in 339).

344. (BUG-STABILITY ISSUES) AsFer-VIRGO Boost::Python system calls invocations analysis - commits - 4 November 2016

Some resumed analysis of AsFer-VIRGO boost::python invocations of VIRGO memory system calls which were stopped few mo
The i915 DRM GEM error is highly reproducible pointing something unruly about it. No logical reason can be attributed
some kernel sync issues. This further confirms that there is nothing wrong with VIRGO layer of Linux kernel and Linu
within inherently has a chronic problem.

345. (THEORY) Polynomial Reconstruction (PR) Problem, Error Correcting Codes and 2-Coloring/Complement Functions - im
draft updates to http://arxiv.org/pdf/1106.4102.pdf - 10 November 2016 and 17 November 2016

Polynomial Reconstruction Problem states that:
Given a set of n points (x1,y1),(x2,y2),...,(xn,yn) recover all polynomials p of degree less than k such that pi(xi)
for maximum of i points in {1,2,3,...,n}. In other words PRP is a curve-fitting interpolation problem and is related
Error correcting problems like List Decoding (list of polynomials approximating a message one of which is correct),Re

2-Coloring scheme algorithms and Complement Function constructions are alternative spectacles to view the Polynomial
Problem i.e Complement Function and 2-Coloring are special settings of Polynomial Reconstruction with exact curve fit

References:

345.1 Polynomial Reconstruction and Cryptanalysis - Berlekamp-Welch, Guruswami-Sudan algorithms - https://eprint.iacr
345.2 Hardness of Constructing Multivariate Polynomials over Finite Fields - [ParikshitGopalan,SubhashKhot,RishiSaket
345.3 Lagrange Interpolation and Berlekamp-Welch PRP algorithm - [SadhkanRuma] - https://www.academia.edu/2756695/Eva

346. (FEATURE-DONE) Major rewrite to Rule Search for Sequence Mined Astronomical Datasets

Commits - 17 November 2016

1. Rule Search script has been updated to include Cusps too to find cross-cusp patterns
2. SequenceMining.py script has been updated specific to NeuronRain AsFer Research version
3. Rule Search script has been rewritten on top of SourceForge revision 945 which was overwritten in later
commits by replicated GitHub version (NeuronRain Enterprise)
4. Logs for above have been committed to testlogs (next commit)

```
5. MinedClassAssociationRules.txt has been updated to latest sequence mining version with Cusps.
6. This commit is specific to NeuronRain Research version in SourceForge and is not replicated in GitHub NeuronRain
Enterprise repository.

---------------------------------------------------------------------------------------------------------------------
347. (THEORY) Data written in electronic storage devices (e.g CD/DVD) and 2-Coloring/Complement Functions - important
draft updates to http://arxiv.org/pdf/1106.4102.pdf - 24 November 2016 and 28 November 2016
---------------------------------------------------------------------------------------------------------------------
CD/DVD storage devices contain binary data burnt on concentric cylindrical tracks. A radial line from central bull's
complete circle in a scan covers 2^n possible binary strings. In the best case all 2^n words are distinct. Total Numb
2^n words are in fibonacci sequence:
        f(n) = 2f(n-1) + 2^(n-1)
with f(0)=0 and f(1)=1. In 2-coloring parlance, DVD is 2-colored with 1(red) and 0(blue) where number of monochromati
bounded by:
        f(n) = 2f(n-1) + 2^(n-1) out of minimum n*2^n possible bits on the storage. Thus any data written to storage
        f(3) = 2f(2) + 2^2 = 2(2f(1) + 2) + 2^2 = 8 + 4 = 12 minimum possible monochromatic bit positions
This has some applications of Hales-Jewitt Theorem for multidimensional 2-coloring - storage device always has a monc
        00
        01
        10
        11
are minimum possible distinct binary words swept by a radial scan of the circle and there are f(2)=2f(1) + 2 = 4 monc

The previous question has been answered in 2.10.  There could be \/ shaped gaps in circular arrangement of 2^n binary
        g(n) = f(n) + f(n-1) + f(n-2) + ... + f(1) + f(0)
Thus g(n) is the tightest bound for number of 1s and 0s. Prefix "minimum" has been added in this bound because repeti

-----------------------------------------------------------------------------------------------
348. (FEATURE-DONE) NeuronRain C++-Python System calls invocation - Commits - 29 November 2016
-----------------------------------------------------------------------------------------------
1. Further analysis of NeuronRain AsFer-VIRGO boost::python system calls invocation - there has been
an erratic -32 and -107 errors
2. But VIRGO system calls work without problems - malloc,set and get work as expected
3. There is a later panic outside VIRGO code but logs have not been found.
4. Logs for this have been added in testlogs/
5. boost C++ code has been rebuilt.

---------------------------------------------------------------------------------------------------------------------
349. (THEORY) Kleinberg Lattice , Random Graph Ontologies, Bose-Einstein model and Recursive Gloss Overlap algorithm
---------------------------------------------------------------------------------------------------------------------
Result of r=2 in [Kleinberg - 202.19] is an equilibrium state. Random shortcut edges are created with some probabilit
        random edge probability(d(n1,n2)) = |l(n1,n2)|^(-r) where l is the lattice distance
When r is less than 2 or close to zero, random edges are as numerous as lattice edges (random edges exist in abundanc

Above small world phenomenon can be mapped to ontology of linguistic concepts/words. Let set of all concepts/words fo

Kleinberg criterion of r=2 is an alternative way to assess the meaningfulness of a document. Document definition grap
        r * log(l(n1,n2)) = log(1/d(n1,n2))
When r=2:
        2log(l(n1,n2)) = log(1/d(n1,n2))
which stipulates conditions for high intrinsic merit for a document in terms of lattice distance and random edge prob

Bose-Einstein model for networks relates fitness of a vertex (ability to attract edges) in a graph and Bose-Einstein

References:
----------
349.1 Bose-Einstein Model and Complex Networks - https://en.wikipedia.org/wiki/Bose%E2%80%93Einstein_condensation_(ne
349.2 Golden mean as a clock cycle of brain waves - http://www.v-weiss.de/chaos.html - "...In 2001 Bianconi and Barab

---------------------------------------------------------------------------------------------------------------------
350. (THEORY) Randomness, Quantum Machine Learning and a Schroedinger Cat simulation of learning patterns in BigData
---------------------------------------------------------------------------------------------------------------------
Caution: This section postulates a drastically new theory mapping quantum mechanics to learning patterns in bigdata w

State of a subatomic particle is defined by a wave function on Hilbert Space (Complex State Vector Space). In Dirac n

In terms of BigData analytics, learning a variable in a blackbox is reducible to Schroedinger's Cat paradox. Assuming

Any BigData set with apparent randomness can be construed as a series of outputs over time generated by a randomized

An alternative formulation of F+G is:
F is the quantum randomness source (e.g Double slit) and G invokes (or measures) F to produce a datastream over time.

There are two levels of learning possible:
*) Learning within blackbox - Learning patterns from observations of F+G. F+G are pre-equipped with ability to learn

*) Learning outside blackbox - This is obvious contradiction mentioned previously because dataset generated by F+G is

Thus above seems to imply learning is impossible with quantum randomness.

References:
```

```
6214 ----------
6215 350.1 Feynman Lectures on Physics - [RichardFeynman] - Volume 3 - Chapter 1 and 2
6216 350.2 Emperor's New Mind - [RogerPenrose] - Chapter 6 - Quantum Magic and Quantum Mystery
6217 350.3 Quantum Randomness and Logical Independence - https://arxiv.org/pdf/0811.4542v2.pdf - Mutually independent logi
6218
6219 ----------------------------------------------------------------------------------------------------------------------
6220 351. (THEORY) Quantum Parallelism, Quantum Interference, Integer Factoring, Periodicity Finding, Polynomial Reconstru
6221 Complement Function/2-Coloring - related to 24,34,323,338,345 and 347 - 29 December 2016 - important draft updates to
6222         - http://sourceforge.net/projects/acadpdrafts/files/DiscreteHyperbolicPolylogarithmicSieveForIntegerFactoriza
6223         - http://arxiv.org/pdf/1106.4102.pdf
6224 ----------------------------------------------------------------------------------------------------------------------
6225 Discrete Hyperbolic Factorization draft in 34 claims (disputed because of input size though references exist to prove
6226        *) Quantum Parallelism in which a function f(x) is computed to get many values of f(x) in parallel
6227        *) Quantum Interference by Hadamard transform where cancellation occurs ( |0> = 1/sqrt(2)*(|0>+|1>) and |1> =
6228
6229 Period Finding in Quantum Computation finds smallest r such that f(x)=f(x+r) and Shor's factorization internally appl
6230
6231 References:
6232 -----------
6233 351.1 Quantum Computation Course notes - Order Finding, Fermat's Little Theorem and Simon, Shor Factorization algorit
6234 351.2 Progress on Quantum Algorithms - [PeterShor] - http://www-math.mit.edu/~shor/papers/Progress.pdf
6235
6236 ----------------------------------------------------------------------------------------------------------
6237 352. (FEATURE-DONE) Recommender Systems Implementation based on ThoughtNet Hypergraph - 1 January 2017
6238 ----------------------------------------------------------------------------------------------------------
6239 *) Input to python-src/DeepLearning_ReinforcementLearningRecommenderSystems.py is a set of observations which are use
6240 (shopping cart items, academic articles read by user etc.,) and already built thoughtnet hypergraph history is search
6241 evocatives from the input
6242 *) Evocatives returned from ThoughtNet hypergraph are items recommended relevant to user's activities.
6243 *) This technique of recommendation is more qualitative than usual methods of Collaborative Filtering and mimicks the
6244 thought process of "relevance based evocation" based on past experience (i.e ThoughtNet history)
6245 *) New input text file and folder RecommenderSystems has been added to python-src with Neo4j Graph Database support
6246 *) Logs have been committed in testlogs/
6247
6248 ----------------------------------------------------------------------------------------------------------
6249 353. (FEATURE-DONE) Kafka data storage for Streaming Abstract Generator - 2 January 2017
6250 ----------------------------------------------------------------------------------------------------------
6251 Kafka Streaming Platform has been added as a data storage in Streaming Generator Abstraction Single-Window entrypoint
6252 code for a neuronraindata topic polls for incoming messages in iterator. As an example Streaming CountMeanMinSketch i
6253 been updated to source streamed data published in Kafka.
6254
6255 ----------------------------------------------------------------------------------------------------------------------
6256 354. (FEATURE-DONE) An example usecase of Recommender Systems for online shopping cart - 2 January 2017
6257 ----------------------------------------------------------------------------------------------------------------------
6258 *) RecommenderSystems folder has been updated with new text file from an example past shopping cart history items (e.
6259 topics in Amazon online bookstore)
6260 *) From the above edges a Hypergraph is created by classifying the shopping cart items into classes and constructing
6261 classes (by Recursive Gloss Overlap Graph maximum core number classifier)
6262 *) Text files for above usecase have .shoppingcart suffixes
6263 *) Above shopping cart hypergraph has a past history of items chosen by a user from variety of topics.
6264 *) New input file RecommenderSystems.shoppingcart.input.txt has present items in user's shopping cart and new items h
6265 user based on present choice and past history.
6266 *) For this, items in present shopping cart are lookedup in past sales history Hypergraph created previously and recc
6267 *) Items in present shopping cart are looked up in two ways: 1) By classifying with Recursive Gloss Overlap graph and
6268 less than a threshold (presently core number 5) and 2) Raw token lookup
6269 *) Logs for these two lookups based Recommendations generated are committed to testlogs/
6270
6271 An important note: Rationale for Hypergraph based past history is that a meaningful text can be classified on multipl
6272 supervised classifiers which classify text on exactly one class. This is more realistic because text can delve into m
6273
6274 ----------------------------------------------------------------------------------------------------------------------
6275 355. (FEATURE-DONE) More detailed shopping cart example 2 for RGO+ThoughtNet based Recommender System - 3 January 201
6276 ----------------------------------------------------------------------------------------------------------------------
6277 *) python-src/DeepLearning_ReinforcementLearningRecommenderSystems.py has been updated to choose top percentile class
6278 so that vertices with large core numbers get more weightage.
6279 *) New shoppingcart2 product reviews text has been added and corresponding Hypergraph history has been created. This
6280 product description with an assorted mix (TV reviews, Washing Machine reviews, Home Theatre reviews etc.,) - python-s
6281 python-src/RecommenderSystems/RecommenderSystems_Hypergraph_Generated.shoppingcart2.txt
6282 *) New input file python-src/RecommenderSystems.shoppingcart2.input.txt has been added with an example product (TV).
6283 *) Recommender System chooses relevant TV products from Hypergraph history and displays to user.
6284 *) Logs have been committed to testlogs/
6285
6286 ----------------------------------------------------------------------------------------------------------------------
6287 356. (THEORY) NEXP not in non-uniform ACC, P(Good) majority voting circuit and some contradictions - 4 January 2017 a
6288 - related to all P(Good) majority voting circuit related points in this document (e.g 14,53 etc.,)
6289 ----------------------------------------------------------------------------------------------------------------------
6290 RHS of P(Good) majority voting circuit has exponential size with unrestricted depth in worst case and is in EXP (if d
6291 result by [RyanWilliams] limits that NEXP is not in ACC (AC circuits with counter mod[m] gates which output 1 if sum
6292 of m). Non uniform ACC is contained in TC and AC is contained in ACC (AC in ACC in TC). This makes a contradiction if
6293 percolation circuit in non-uniform NC (NC/poly). If P(Good) binomial coefficient series summation converges to 100%,
6294 more counterexample implying some or all of the following:
```

```
6295        *) There is no 100% efficient RHS majority voting boolean function composition
6296        *) There is no 100% efficient LHS boolean function (pseudorandomly chosen boolean function, ranked by social c
6297        *) RHS Majority voting is not in NEXP.
6298        *) LHS pseudorandom choice boolean cannot have a non-uniform NC circuit
6299  Above, "efficiency" implies "No error" cases in scenarios matrix of 53.14 and a ninth error scenario of Randomized De
6300  ------------------------------------------------------------------------------------------
6301         x                              |    f(x) = f(x/e)      |       f(x) != f(x/e) Noise    |
6302  ------------------------------------------------------------------------------------------
6303  x in L, x/e in L                      |       No error        |       Error                   |
6304  ------------------------------------------------------------------------------------------
6305  x in L, x/e not in L                  |       Error           |   No error if f(x)=1,f(x/e)=0 |
6306                                        |                       |   else Error                  |
6307  ------------------------------------------------------------------------------------------
6308  x not in L, x/e in L                  |       Error           |   No error if f(x)=0,f(x/e)=1 |
6309                                        |                       |   else Error                  |
6310  ------------------------------------------------------------------------------------------
6311  x not in L, x/e not in L              |       No error        |      Error                    |
6312  ------------------------------------------------------------------------------------------
6313  ------------------------------------------------------------------------------------------
6314        x                               |                      f(x)                          |
6315  ------------------------------------------------------------------------------------------
6316  Randomized Decision tree evaluation   |       No error        |       Error                   |
6317  ------------------------------------------------------------------------------------------
6318  If LHS social choice ranking function is Interview algorithm which is a PSPACE=IP algorithm (by straightforward reduc
6319
6320  There are natural processes like Soap Bubble formation known to solve Steiner Tree NP-hard problem efficiently. Soap
6321  glass plates are formed and bubbles are connected by line segments of optimum total length which converge at Steiner
6322  also a natural process with human element involved. Does human judgement overwhelm algorithmic judgement in decision
6323  open question. If neural networks are algorithmic equivalents of human reasoning, then each voter decision function c
6324  non-uniform TC neural network. Error of majority voting is then equal to error of this majority function + neural net
6325
6326  Previous matrix of 10 scenarios basically subdivides the traditional BP* definition which says: For x in L, a BP* tur
6327         *) (f(x)=f(x/e)) in which case two correlated strings one in L and other not in L are both accepted (false po
6328         *) (f(x) != f(x/e) - Noise sensitivity) in which case two correlated strings in L or not in L are erroneously
6329         *) Previous two are input related while the last scenario with error in decision tree evaluation is internal
6330
6331  This false positive + false negative voter judgement error applies to BPTC NC+neural network circuit composition alsc
6332
6333  References:
6334  -----------
6335  356.1 NEXP not in non-uniform ACC - [RyanWilliams] - http://www.cs.cmu.edu/~ryanw/acc-lbs.pdf
6336  356.2 Soap Bubble Steiner Tree and P=NP - [ScottAaronson] - https://arxiv.org/pdf/quant-ph/0502072v2.pdf
6337  356.3 Constant depth Threshold circuits - http://people.cs.uchicago.edu/~razborov/files/helsinki.pdf
6338  356.4 Circuit Complexity of Neural networks - https://papers.nips.cc/paper/354-on-the-circuit-complexity-of-neural-ne
6339  356.5 Exact Threshold circuits - http://www.cs.au.dk/~arnsfelt/Papers/exactcircuits.pdf
6340  356.6 Universal Approximation Theorem - [Cybenko] - https://en.wikipedia.org/wiki/Universal_approximation_theorem - M
6341  356.7 L* algorithm for exact learning of DFAs - [Dana Angluin] - https://people.eecs.berkeley.edu/~dawnsong/teaching/
6342  356.8 Efficient Learning Algorithms Yield Circuit Lower Bounds - [Lance Fortnow] - http://lance.fortnow.com/papers/fi
6343
6344  ------------------------------------------------------------------------------------------------------------------------
6345  357. (FEATURE-DONE) PythonSpark+Cython Interview Algorithm cloud implementation update - 4 January 2017 - related to
6346  ------------------------------------------------------------------------------------------------------------------------
6347  PythonSpark+Cython Cloud Implementation of Interview Algorithm has been updated to print average clustering coefficie
6348  graph for text. Average clustering coefficient is average(per node clustering coefficient) for all nodes where cluste
6349  a vertex is a ratio of edges to neighbours to all possible edges to neighbours (or) amount of "cliqueness" of each ne
6350  vertices in a graph. This adds an alternative way for computing intrinsic merit of a document recursive gloss overlap
6351  349 (Kleinberg's small world graph and clustering coeffient = 2). Average clustering coefficient supplements the intr
6352  score in deciding semantic relatedness of a graph. Closer the clustering coefficient is to 2, it is easier to find pa
6353  and greater the linguistic meaningfulness.
6354
6355  ------------------------------------------------------------------------------------------------------------------------
6356  358. (THEORY) P(Good) non-majority versus majority social choices, BPTC and PSPACE classes - related to 317,356 - 6 J
6357  ------------------------------------------------------------------------------------------------------------------------
6358  Caution: Following tries to prove a major lowerbound result with some assumptions and is subject to errors.
6359
6360  Main motivation for so much emphasis on P(Good) binomial summation convergence is: LHS is non-majority social choice
6361  social choice and convergence to 100% on both sides implies LHS and RHS are of equal merit with varying complexity cl
6362  choice process is as follows:
6363
6364  foreach(voter)
6365  {
6366        Interview the voter (a PSPACE-complete problem where polynomial number of question-answering reduces to prover
6367  }
6368  Rank the voters by merit and choose the topmost as non-majority social choice - voting is obviated.
6369
6370  It is interesting to note that web search engines use both non-majority (Ranking by merit) and majority choice (e.g F
6371
6372  ------------------------------------------------------------------------------------------------------------------------
6373  359. (THEORY) KRW Conjecture on Boolean Function Composition, Non-majority (PSPACE) and Majority (harder than PSPACE)
6374  related to 14,53,311,356 - 7 January 2017
6375  ------------------------------------------------------------------------------------------------------------------------
```

```
Previous point mooted the idea of the parallel interview of voters and ranking them as a non-majority social choice.
the interview algorithm scores of individual voters and finding topmost. Sorting of numbers is in AC=NC=TC (Sorting n
 [ChandraStockMeyerVishkin] Constant Depth Reducibility for Sorting - http://www.cstheory.com/stockmeyer@sbcglobal.ne
Threshold circuits for sorting) and each voter's PSPACE-complete interview circuit is input to NC sorting circuit to

This creates a following intriguing possibility: If RHS majority voting has voter functions harder than PSPACE (i.e E

References:
-----------
359.1 Karchmer-Raz-Wigderson (KRW) Conjecture of Boolean Function Composition and Information Complexity - http://cs.

------------------------------------------------------------------------------------------------------------------------
360. (THEORY) Non-majority social choices - Interview Ranking Function and Pseudorandom Choice Function - 11 January
317, 359 and all other non-majority versus majority voting points
------------------------------------------------------------------------------------------------------------------------
Previous sections mentioned about two possible ways of non-majority social choice:
       *) Parallel interview of voters and ranking based on sorted interview scores
       *) Psuedorandom choice

------------------
Interview circuit:
------------------
Error in interview circuit which is an NC sorting network with PSPACE-complete voter interviews as inputs is defined
False positives + False negatives + False questions = Percentage of wrong answers marked as right + Percentage of rig

For sorting purposes output of TQBF is not a binary 0 or 1 but a binary string with value equal to number of correct
other boolean functions sensitivity of TQBF phi(q1,a1,q2,a2,q3,a3,...,qn,an) is defined as how flipping of ai and qi
Formally,
       Sensitivity(TQBF) = Pr(TQBF(X) != TQBF(Xcorrelated)) where Xcorrelated is an interview with erroneous question

Thus sensitivity captures the error in interview. It has to be noted that wrong questions also measure the flaw in in

--------------------
Pseudorandom Choice:
--------------------
Set of all voter decision functions is partitioned into n sets where each set has goodness xi. Here goodness of a vot
is defined as: 1-error(f).

Let number of voter decision functions with goodness xi = m(xi). Thus N = m(x1)+m(x2)+m(x3)+...+m(xn)
Expected goodness of a PRG choice is:
       1/N * summation(xi*m(xi)) = (x1*m(x1) + x2*m(x2) + x3*m(x3) + ... + xn*m(xn)) / N
When all voter functions have goodness 1 then PRG choice in LHS of P(good) has goodness 1.

These two non-majority choices are just hypothetical examples of how a social choice can be made without voting. In r
social choice occurs is quite complex and determined by various factors of society ( economic disparities, ethnicity

------------------------------------------------------------------------------------------------------------------------
361. (FEATURE-DONE) Commits - 11 January 2017 - DeepLearning Convolution Networks update
------------------------------------------------------------------------------------------------------------------------
Some changes done to Convolution computation:
*) Made sigmoid perceptron optional in final neural network from Maxpooling layer so that weighted sum is printed ins
*) This causes the convolution network to be very sensitive to presence of pattern
*) 2 more example bitmaps have been added with varying degree of pattern prominence (a pronounced X and a thin 1)
*) With this final neurons from maxpooling layer print values of neurons which quite closely reflect the pattern's ma
results in big value while small pattern results in small value proportionately.
*) This can rank the bitmaps in increasing degree of magnitude of pattern presence
*) Logs for this have been committed to testlogs

------------------------------------------------------------------------------------------------------------------------
362. (FEATURE-DONE) DeepLearning Convolution Network - an example pattern recognition from bitmap images - 12 January
------------------------------------------------------------------------------------------------------------------------
*) Few more bitmap images have been included with same pattern but of different sizes
*) There are 5 patterns and 9 bitmaps:
        - Thin X and Boldfaced X
        - Thin 0 and Boldfaced 0
        - Thin 8 and Boldfaced 8
        - No pattern
        - Thin 1 and Boldfaced 1
*) Expectation is that Convolution Network final neuron layer must output similar values for same pattern and differe
for different patterns
*) pooling_neuron_weight has been reintroduced in final neuron layer of 10 neurons each with a randomly chosen weight
*) Logs show for all 10 neurons, final neural activation values are close enough for similar patterns and different f
*) Thus Convolution Network which is a recent advance in DeepLearning works quite well to recognize similar image pat

In above example , 9th and 10th neurons output following values. Example 11 and 12 are similar patterns - 0 and 0. Ex
##############################################################################################
Inference from Max Pooling Layer - Neuron  8
##############################################################################################
Example 11:
###########
[27.402311485751664]
```

```
6457  ##########
6458  Example 12:
6459  ##########
6460  [27.36740767846171]
6461  ##########
6462  Example 21:
6463  ##########
6464  [32.15200939997122]
6465  ##########
6466  Example 22:
6467  ##########
6468  [30.835940458495205]
6469  ##########
6470  Example 3:
6471  ##########
6472  [17.133246549473675]
6473  ##########
6474  Example 41:
6475  ##########
6476  [29.367605168715038]
6477  ##########
6478  Example 42:
6479  ##########
6480  [27.446193773968886]
6481  ##########
6482  Example 51:
6483  ##########
6484  [22.12145242997834]
6485  ##########
6486  Example 52:
6487  ##########
6488  [24.17603022706531]
6489  ##############################################################################################
6490  Inference from Max Pooling Layer - Neuron  9
6491  ##############################################################################################
6492  Example 11:
6493  ##########
6494  [24.667080337176497]
6495  ##########
6496  Example 12:
6497  ##########
6498  [24.635666910615544]
6499  ##########
6500  Example 21:
6501  ##########
6502  [28.941808459974094]
6503  ##########
6504  Example 22:
6505  ##########
6506  [27.757346412645685]
6507  ##########
6508  Example 3:
6509  ##########
6510  [15.424921894526316]
6511  ##########
6512  Example 41:
6513  ##########
6514  [26.435844651843524]
6515  ##########
6516  Example 42:
6517  ##########
6518  [24.70657439657199]
6519  ##########
6520  Example 51:
6521  ##########
6522  [19.914307186980512]
6523  ##########
6524  Example 52:
6525  ##########
6526  [21.763427204358788]
6527
6528  ----------------------------------------------------------------------------------------------
6529  363. (FEATURE-DONE) DeepLearning BackPropagation Implementation Update - 17 January 2017
6530  ----------------------------------------------------------------------------------------------
6531  *) DeepLearning BackPropagation code has been changed to have 3 inputs, 3 hidden and 3 output layers
6532  with 3*3=9 input-hidden weights and 3*3=9 hidden-output weights with total of 18 weights
6533  *) Software Analytics example has been updated with a third input and logs for it have been committed
6534  to testlogs/.
6535  *) Logs include a diff notes of how to extend this for arbitrary inputs and accuracy of backpropagation
6536  which beautifully converges at ~10^-26 error after ~1000000 iterations.
6537
```

```
6538 ----------------------------------------------------------------------------------------------------
6539 364. (FEATURE-DONE) Software Analytics with DeepLearning - 18 January 2017
6540 ----------------------------------------------------------------------------------------------------
6541 *) An example software analytics code based on DeepLearning implementation has been added to
6542 software_analytics which import BackPropagation,Convolution and RecurrentLSTM to learn models
6543 from software analytics input variables (CPU%, Memory% and TimeDuration%)
6544 *) Logs for all 3 models learnt with same input variables have been added to software_analytics/testlogs
6545
6546 ----------------------------------------------------------------------------------------------------------------------
6547 365. (THEORY and IMPLEMENTATION) Polynomial Encoding of a Text and 2 new distance measures based on it - related to 2
6548 - 19 January 2017
6549 ----------------------------------------------------------------------------------------------------------------------
6550 *) It has been earlier mentioned that an alphanumeric text can be construed as a polynomial defined as f:position->al
6551 *) This python implementation encodes a text as a polynomial by applying NumPy Polyfit function on ordinal values of
6552 letter positions as y-axis and letter positions as x-axis.
6553 *) A sample polynomial plotted with matplotlib has been added to testlogs/ alongwith logs
6554 *) Once a text is plotted as a polynomial curve, it is natural to define distance between two strings as distance
6555 between two respective polynomial encodings.
6556 *) Usual distance measures for strings e.g Edit distance are somewhat qualitative and do not quantify in numeric term
6557 *) Distance between polynomials of two texts is numerically quite sensitive to small changes in texts and visually ma
6558 definition of "distance" between two polynomials of strings.
6559
6560 There are two distance functions defined for polynomial representation of texts:
6561 *) Distance between polynomials is defined by inner product of polynomials f(x) and g(x) and in discrete version is t
6562 L2 norm : sum(i(f(xi)-g(xi))^2)
6563 *) Distance between polynomials considering them as two discrete set of probability distributions - Kullback-Leibler
6564  Divergence measures.
6565
6566 ----------------------------------------------------------------------------------------------------------------------
6567 For example following are two polynomials fitted for ordinal values of 2 texts and comparative distance values are pr
6568 edit distance and 2 polynomial distance measures:
6569 text9="fdjfjkkkkkkkkkkkkkfjjjjjjjjjjjskwwwwwwwwwwwwwwwwwwiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii"
6570 text10="wjejwejwkjekwjkejkwjekjwkjekwjjoisjdiwidoiweiwie0iw0eio0wie0wie0iw0ei0cndknfndnfndnfkdjfkjdfkjd
6571 ----------------------------------------------------------------------------------------------------------------------
6572 X-axis: xrange(94)
6573 Y-axis: [102, 100, 106, 102, 106, 107, 107, 107, 107, 107, 107, 107, 107, 107, 107, 107, 107, 107, 102, 106, 106, 106
6574 Coefficients of fitting polynomial of degree 5: [  1.07744279e+02  -1.44951431e+00   1.33928005e-01  -3.69850703e-03
6575    4.03119671e-05  -1.53297932e-07]
6576 X-axis: xrange(95)
6577 Y-axis: [119, 106, 101, 106, 119, 101, 106, 119, 107, 106, 101, 107, 119, 106, 107, 101, 106, 107, 119, 106, 101, 107
6578 Coefficients of fitting polynomial of degree 5: [  1.13816036e+02  -2.17630157e+00   1.96357195e-01  -6.45349037e-03
6579    8.34267469e-05  -3.67775407e-07]
6580 ...
6581 ################################################################################
6582 Two text polynomial ordinal points are represented as set of ordered pairs
6583 ################################################################################
6584 Wagner-Fischer Edit Distance for text9 and text10: 76
6585 Polynomial Encoding Edit Distance (JensenShannon) for text9 and text10:
6586 str1str2_tuple: [(102, 119), (100, 106), (106, 101), (102, 106), (106, 119), (107, 101), (107, 106), (107, 119), (107
6587 421.059610023
6588 Polynomial Encoding Edit Distance (L2 Norm) for text9 and text10:
6589 str1str2_tuple: [(102, 119), (100, 106), (106, 101), (102, 106), (106, 119), (107, 101), (107, 106), (107, 119), (107
6590 200.743617582
6591 ################################################################################
6592 From above, usual edit distance between 2 strings is less than polynomial distances of JensenShannon and L2 norms. Or
6593
6594 ----------------------------------------------------------------------------------------------------------------------
6595 366. (THEORY) BKS conjecture and Stability of Interview TQBF - related to 360 - 21 January 2017
6596 ----------------------------------------------------------------------------------------------------------------------
6597 Caution: This derivation is still experimental with possible errors.
6598
6599 Stability of a boolean function is defined as:
6600         Stability(f) = Expectation(f(x)*f(y))
6601 where y is correlated version of x.
6602
6603 Stability of Majority is defined as (From page 120 of AnalysisOfBooleanFunctions - Chapter 5 Majority and Threshold F
6604         Stability(Majn) = Expectation(Majn(x)*Majn(y))
6605 where y is correlated version of x.
6606
6607 For infinite n, Stability(Majn) is bounded by 2/pi*arcsin(rho). Similar stability measure can be derived for Intervie
6608         w1*a1 + w2*a2 + ... + wn*an > threshold_cutoff
6609 were each ai is an answer for a question qi and wi is the weightage associated with it. Berry-Esseen Central Limit Th
6610
6611 Stability of Interview can be equivalently defined as:
6612         Stability(Interview) = Expectation(Interview(x)*Interview(y))
6613 where y is correlated version of x.
6614
6615 Substituting the threshold definition of interview in Stability:
6616         Stability(Interview) = Expectation((w1*a1+w2*a2+...+wn*an)*(z1*b1+z2*b2+...+zn*bn))
6617                              = Expectation(w1*z1*a1*b1 + w1*z1*a2*b2 + ... + wn*zn*an*bn)
6618                              = a1*b1*Expectation(w1*z1) + .... + an*bn*Expectation(wn*zn)
```

```
6619
6620  Assuming for all ai=bi=a:
6621          Stability(Interview) = a^2 * (Expectation(w1*z1) + Expectation(w1*z2) + .... + Expectation(wn*zn))
6622
6623  Expectation(wi*zk) = 0*0*1/4 + 0*1*1/4 + 1*0*1/4 + 1*1*1/4 for 4 possible values of (wi,zk) each with probability 1/4
6624
6625  =>      Stability(Interview) = a^2 * (n^2 * 0.25) = a^2*n^2/4
6626
6627          Stability(Interview) = a^2*n^2/4 < 1 => a < 2/n
6628
6629  If Stability(Interview) > Stability(Majority), BKS conjecture is true.
6630
6631  =>      a^2*n^2 / 4 > 1 - 2/pi
6632  =>      a > 2/n * sqrt(1-2/pi)
6633  =>      a > 1.2056205488/n [For minimum case of n=2, a > 0.6028102744]
6634
6635  Thus for Stability(Interview) to exceed Stability(Majority) and BKS Conjecture to be true , weight per answer has to
6636
6637  Stability is alternatively defined as:
6638          (+1)*Pr(f(x) = f(y)) + (-1)*Pr(f(x) != f(y))
6639  which is just expansion of Expectation and there are two random variables (+1 for f(x)=f(y)) and (-1 for f(x) != f(y)
6640  => Stability(f(x)) = 1-2*Pr(f(x) != f(y))
6641  => Stability(f(x)) = 1-2*Sensitivity(f(x))
6642  => Sensitivity(f(x)) = 0.5-0.5*Stability(f(x))
6643
6644  NoiseSensitivity(Interview) which is the dual of Stability is defined as:
6645          NoiseSensitivity(Interview) = 0.5-0.5*Stability(Interview) = 0.5-0.5*a^2*n^2/4
6646
6647  ------------------------------------------------------------------------------------------------------------------------
6648  367. (FEATURE-DONE)  Commits - 25 January 2017 - related to 2.9 and 345
6649  ------------------------------------------------------------------------------------------------------------------------
6650  Berlekamp-Welch Polynomial Reconstruction Decoder Implementation (for Reed-Solomon codes, texts, numerical points etc
6651  ------------------------------------------------------------------------------------------------------------------------
6652  *) Implements Berlekamp-Welch Polynomial Reconstruction Algorithm for reconstructing data from errors.
6653  *) System of linear equations has been solved with NumPy/SciPy Linear Algebra solve().
6654  *) Logs show how well the reconstructed polynomial closely approximates the original polynomial.
6655  *) Error locator polynomial E is degree 1 and Numerator Q is of degree = number of points thus yielding Q/E=P (recons
6656  *) Special case is natural language text with errors and a polynomial can be reconstructed from ordinal values of uni
6657
6658  ------------------------------------------------------------------------------------------------------------------------
6659  368. (FEATURE-DONE) Berlekamp-Welch Algorithm implementation - update for text message decoding - 27 January 2017
6660  ------------------------------------------------------------------------------------------------------------------------
6661  *) Using unicode values for ordinals with ord() and inverting with chr() causes overflow errors in SciPy/NumPy
6662  Linear Algebra Equation solver (solve()) because evaluating polynomials for large messages creates huge numbers
6663  *) Hence unicode has been replaced with a simple dictionaries - alphanumeric to numeric values and numeric to
6664  alphanumeric values.
6665  *) With above 2 dictionaries an example text message with error (garbled version of "thisissentence") is list
6666  decoded with a window of possible values for each letter positions.
6667  *) Logs show an approximate decoding of original message from message with error.
6668  *) Garbled text with error was created from a previous execution of Berlekamp-Welch algorithm.
6669
6670  ------------------------------------------------------------------------------------------------------------------------
6671  369. (THEORY) KRW Conjecture, KW relations, Majority Voting Circuit Composition - 30 January 2017 and 31 January 2017
6672  ------------------------------------------------------------------------------------------------------------------------
6673  Majority Voting Circuit for P(Good) binomial series RHS is a boolean circuit composition of NC Majority voting circui
6674  Voter Decision Functions for each voter variable input to Majority function defined as:
6675           Maj(m) + Voter(n) = Maj(Voter1(x1,x2,...,xn),Voter2(x1,x2,...,xn),...,Voterm(x1,x2,...,xn)).
6676  Assumption is Voter Decision Functions can be exactly learnt in Angluin Exact Learning model i.e Voters can have zero
6677  KRW Conjecture for boolean formula composition implies Depth(f+g) ~ Depth(f) + Depth(g) for composition f+g of two bo
6678  (fanout 1) f and g. Applying KRW conjecture to Maj(m) + Voter(n) Majority voting composition:
6679          Depth(Maj(m) + Voter(n)) ~ Depth(Maj(m)) + Depth(Voter(n))
6680  where m is the number of voters and n is the number of variables per voter decision function.
6681
6682  KW relations R(f) for a function f imply Depth(f) = CommunicationComplexity(R(f)).
6683  Karchmer-Wigderson relations for the composition are the following:
6684  -------------
6685  R(Maj,Voter):
6686  -------------
6687          Alice: x in Inverse(Maj + Voter)(0)
6688          Bob:  y in Inverse(Maj + Voter)(1)
6689          Find xi and yi such that x != y
6690
6691  Depth(Maj + Voter) = CommunicationComplexity(R(Maj,Voter))
6692  Thus Majority voting circuit composition is a KW relation.
6693  There is an equivalent notion of Universal relation U(n) for KW relation defined as:
6694          Alice: x in {0,1}^(n)
6695          Bob:  y in {0,1}^(n)
6696          Find xi and yi such that x != y
6697
6698  Result in 369.2 prove a lowerbound:
6699          Depth(g+U(n)) = CommunicationComplexity(R(g,U(n))) >= log L(g) + n - O(m*logm/n) where L(g) is the formula si
```

```
If g is Majority function and U(n) is individual voter decision function, depth of composition of Majority and Voter
       D(Maj + Voter) = CommunicationComplexity(R(Maj,Voter)) >= log L(Majority) + n - O(m*logm/n)
But formula size of Majority is O(m^5.3) implying depth of Majority+VoterDecisionFunction composition is:
       D(Maj + Voter) = CommunicationComplexity(R(Maj,Voter)) >= 5.3*logm  + n - O(m*logm/n)
Thus depth of Majority voting circuit for P(Good) RHS is lowerbounded by a function of number of voters and number of

369.1 prove a result which is an extension of 356.8:
If a class C is exactly learnable in polynomial time, DTIME(n^omega(1)) is not in C. In mistake bounded learning mode

Exact Learning implies zero error on both sides of P(Good) binomial series. Assuming all boolean function are exactly

       Let number of  voter decision functions with goodness xi = m(xi). Thus N = m(x1)+m(x2)+m(x3)+...+m(xn)
       Expected goodness of a PRG choice is:
              1/N * summation(xi*m(xi)) = (x1*m(x1) + x2*m(x2) + x3*m(x3) + ... + xn*m(xn)) / N
       When all voter functions have goodness 1 then PRG choice in LHS of P(good) has goodness 1.

Similarly, exact learning implies all voter decision functions are 100% perfect with zero-error in LHS. Binomial seri

Usual process of high level human learning vis-a-vis boolean function learning is bottom-up. Atomic concepts are lear

Hypothetical exact learner for 3SAT - NP voter decision functions :
-------------------------------------------------------------
Given a truth table of size n*2^n for n variables with truth values:
       *) Find a set of binary strings, satisfying truth values 1
       *) Construct a DNF for complement of the previous set
       *) Complement the DNF to get CNF for original set

Example - For following truth table, to construct CNF for satisfying assignments {000, 001, 011, 101, 110}:
       ---------------------------------------
       x1 x2 x3        truthvalue      complement
       ---------------------------------------
       0  0  0         1               0
       0  0  1         1               0
       0  1  0         0               1
       0  1  1         1               0
       1  0  0         0               1
       1  0  1         1               0
       1  1  0         1               0
       1  1  1         0               1

DNF for complement set {010,100,111} is:
       !x1x2!x3 + x1!x2!x3 + x1x2x3
and the CNF learnt is complement of DNF:
       (x1 + !x2 + x3)(!x1 + x2 + x3)(!x1 + !x2 + !x3)
with satisfying assignments {000, 001, 011, 101, 110}

This exact learner requires O(2^n) time which is exponential in number of variables.

369.6 describes an algorithm (CDNF) to learn any boolean functions in time polynomial in number of variables,its DNF

References:
----------
369.1 Exact Learning implies Derandomization (i.e. P=BPP) and Mistake bound learning - [KlivansKothariOliveira] - htt
369.2 KRW Conjecture, KW relations and Lowerbound for Depth of Boolean Circuit Composition - [GavinskyMeirWeinsteinWi
369.3 Generalization of Spira's Theorem - https://www.cs.utexas.edu/users/panni/spira.pdf - Spira's theorem transform
369.4 Depth Hierarchy Theorem - [RossmanServedioTan] - Theorem 1 - https://arxiv.org/pdf/1504.03398v1.pdf - Circuit c
369.5 Formula size of Majority function - [Valiant] - http://www.sciencedirect.com/science/article/pii/01966774849001
369.6 Exact Learning of Boolean Functions from Monotone Theory - [Nader H. Bshouty] - http://www.cs.technion.ac.il/~b
369.7 BPEXP is believed to collapse to EXP - if P=BPP then BPEXP=EXP (but converse BPEXP=EXP implying P=BPP is not kn
369.8 From [Klivans-Fortnow] result mentioned in 369.1, if class EXP is efficiently PAC learnable with membership que
contradicting BPEXP=EXP by homogeneous voter Condorcet Jury Theorem BPEXP circuit derandomization mentioned in 401. B

-----------------------------------------------------------------------------------------------------------------
370. (FEATURE-DONE) An experimental special case k-CNF SAT Solver algorithm implementation - 31 January 2017
-----------------------------------------------------------------------------------------------------------------
Algorithm:
----------
*) Create a k-DNF from k-CNF input by negating literals in all clauses of k-CNF
*) Binary encode the clauses of k-DNF into set of binary strings of length n where n is number of variables
*) Compute the complement of the previous binary encoded set from set of 2^n binary strings
*) These are satisfying assignments
*) Presently requires all literals in each CNF clause
*) Implemented for simulation of a Majority Voting with Voter SATs

-----------------------------------------------------------------------------------------------------------------
371. (THEORY) Majority Boolean Function, Sensitivity, Roth's estimate and 2-Coloring/Complement Functions of Sequence
1 February 2017 - related to 24,338 - Important draft updates to https://arxiv.org/pdf/1106.4102v1
-----------------------------------------------------------------------------------------------------------------
Boolean Majority Function finds the majority binary value of input binary string of length n. Any binary string of le
considered as 2-coloring of bit position integers with corresponding complement functions for each color: f(x) is for
g(x) is for 0s colored blue. For example, 1001011 is a 2-colored sequence with red (1) in bit positions 1,4,6,7 and b
```

```
2,3,5. Complement functions for this bit position binary coloring are defined as:
        f(1)=1
        f(2)=4
        f(3)=6
        f(4)=7
and
        g(1)=2
        g(2)=3
        g(3)=5
Natually, arithmetic progressions and monochromatic arithmetic progressions on these bit positions can be defined. Di
 difference between number of colored integers in an arithmetic progression. Roth's estimate which is the minimum(max
for all possible colorings (i.e all possible binary strings) is >= N^(1/4+epsilon). For majority function 2-coloring,
the difference between number of 0s and 1s in the input to majority function with probability equal to natural densit
        |f(x)| = number of 1s = [N + N^(1/4+epsilon)]/2  if majority function outputs 1
        |g(x)| = number of 0s = [N - N^(1/4+epsilon)]/2  if majority function outputs 1
and vice versa if majority function outputs 0. For previous example, majority is 1 and :
        |f(x)| = number of 1s = [7 + 7^0.25]/2 = 4.313288 ~ 4
        |g(x)| = number of 0s = [7 - 7^0.25]/2 = 2.6867 ~ 3
Sensitivity of Majority function is number of bits to be flipped to change the outcome which is nothing but the discr

------------------------------------------------------------------------------------------------------------------------
372. (FEATURE-DONE) Reinforcement Learning - ThoughtNet based Recommender Systems Implementation Update - 2 February
------------------------------------------------------------------------------------------------------------------------
*) Recursive Gloss Overlap classifier disambiguation has been updated to invoke NLTK lesk algorithm function() with c
choose between PyWSD lesk implementation and best_matching_synset() primitive disambiguation native implementation
*) New usecase shopping cart example with book reviews has been added
*) Fixed a major bug in RecursiveGlossOverlap Classifier in definition graph construction : Edges for all lemma names

------------------------------------------------------------------------------------------------------------------------
373. (FEATURE-DONE) Recursive Gloss Overlap classifier update - 3 February 2017
------------------------------------------------------------------------------------------------------------------------
Uncommented NetworkX matplotlib graph plotting to get a graph for a sample document definition graph. Logs and PNG fi

------------------------------------------------------------------------------------------------------------
374. (THEORY and FEATURE-DONE) CNFSAT solver update - polynomial time approximation - 5 February 2017
------------------------------------------------------------------------------------------------------------
Solves CNFSAT by a Polynomial Time Approximation scheme:
       - Encode each clause as a linear equation in n variables: missing variables and negated variables are 0, other
       - Solve previous system of equations by least squares algorithm to fit a line
       - Variable value above 0.5 is set to 1 and less than 0.5 is set to 0
       - Rounded of assignment array satisfies the CNFSAT with high probability
Essentially each CNF is represented as a matrix A and solved by AX=B where X is the column vector of variables and B

------------------------------------------------------------------------------------------------------------
375. (FEATURE-DONE) Java Spark Streaming - Generic Stream Receiver Implementation - 6 February 2017
------------------------------------------------------------------------------------------------------------
Spark Streaming Java implementation to receive generic stream of unstructured text data from any URL and Sockets has
NeuronRain AsFer java/bigdata_analytics/. It is based on Spart 2.1.0 + Hadoop 2.7 single node cluster. Following are

*) Oracle Java 8 compiler and runtime and PATH set to it
*) export CLASSPATH=/home/shrinivaasanka/spark-2.1.0-bin-hadoop2.7/jars/spark-streaming_2.11-2.1.0.jar:.:/home/shrini
*) Compiled as: javac SparkGenericStreaming.java
*) JAR packaging of SparkGenericStreaming*.class :  jar -cvf sparkgenericstreaming.jar *class
*) Example Spark submit commandline for 2 local threads (2 cores) and data streamed from twitter :
/home/shrinivaasanka/spark-2.1.0-bin-hadoop2.7/bin/spark-submit --class SparkGenericStreaming --master local[2] spark

Spark Streaming Java implementation complements Streaming Abstract Generator Python native abstraction implemented in

------------------------------------------------------------------------------------------------------------------------
376. (FEATURE-DONE) Java Spark Streaming for Generic data - Jsoup ETL integration - 7 February 2017
------------------------------------------------------------------------------------------------------------------------
*) Jsoup HTML parser has been imported into SparkGenericStreaming.java and a very basic Extract-Transform-Load is
performed on the URL passed in with a special boolean flag to use Jsoup
*) Jsoup connects to URL and does a RESTful GET of the HTML, extracts text from HTML and stores the text in Spark RDD
*) receive() is periodically invoked and word count is computed
*) foreachRDD() is invoked on the DStream to iterate through all words in DStream.
*) forEach() invokes a lambda function to print the word text
*) Logs for this have been added to spark_streaming/testlogs/

spark-submit commandline requires additional classpath to jsoup .jar with --jars option as below:
/home/shrinivaasanka/spark-2.1.0-bin-hadoop2.7/bin/spark-submit --class SparkGenericStreaming --master local[2] --jar

------------------------------------------------------------------------------------------------------------------------
377. (THEORY and FEATURE-DONE) Approximate SAT solver update - 8 February 2017 - related to 374
------------------------------------------------------------------------------------------------------------------------
Updated CNFSATSolver.py to create set of random 3CNFs and verify if the least squares assignment is satisfied. Presen

References:
----------
377.1 PCP theorem and Hardness of Approximation - http://www.cs.jhu.edu/~scheideler/courses/600.471_S05/lecture_9.pdf
377.2 PCP theorem and Hardness of Approximation - http://pages.cs.wisc.edu/~dieter/Courses/2008s-CS880/Scribes/lectur
```

```
6862
6863  ----------------------------------------------------------------------------------------------------------------
6864  378. (FEATURE-DONE) Java Spark Streaming Generic Receiver update - 9 February 2017
6865  ----------------------------------------------------------------------------------------------------------------
6866  *) Objectified the SparkGenericStreaming more by moving the SparkConf and JavaStreamingContext into
6867  class private static data.
6868  *) new method SparkGenericStreamingMain() instantiates the Spark Context and returns the JavaPairDStream<String,Integ
6869  *) main() static method instantiates SparkGenericStreaming class and receives wordCounts JavaPairDStream from SparkGe
6870  *) Spark JavaStreamingContext output operations are invoked by start() and print actions by lambda expressions.
6871  *) SparkConf and JavaStreamingContext variables have to be static because Not Serializable exceptions are thrown othe
6872  *) sparkgenericstreaming.jar has been repackaged by recompilation.
6873
6874  ----------------------------------------------------------------------------------------------------------------
6875  379. (FEATURE-DONE) Spark Streaming Java Update - Java Bean DataFrame(Dataset) creation, Hive Metastore support and p
6876  DataFrame to Hive and as a Parquet file - 10 February 2017
6877  ----------------------------------------------------------------------------------------------------------------
6878  *) New JavaSparkSingletonInstance class for singleton Spark Context has been added.
6879  *) New Java Bean Word.java has been added for creating DataFrame from JavaRDD iterable
6880  *) DataFrame has been persisted to filesystem as .parquet file in overwrite mode of DataFrameWriter
6881  *) DataFrame has also been persisted to Hive MetaStore db which Spark supports (Shark SQL) with saveAsTable()
6882  *) SparkGenericStreaming has been rewritten to do transformations on words JavaRDD iterable with VoidFunction() in
6883  2-level nested iterations of foreachRDD() and forEach()
6884  *) map() transformation for each row in JavaRDD throws a null pointer exception and inner call() is never invoked.
6885  *) This has been remedied by replacing map() with a forEach() and storing the rows in each RDD in an ArrayList
6886  *) With Hive MetaStore + Shark integration, Streaming_AbstractGenerator.py abstraction now has access to Java Spark S
6887  because it already supports Hive via thrift and NeuronRain now has ability to analyze any realtime or batch unstructu
6888  *) Primitive ETL to just scrape words in URL text sufficiently tracks trending topics in social media.
6889  *) Hive metastore db, Spark warehouse data have been committed and Logs for this has been added to testlogs/
6890  *) sparkgenericstreaming.jar has been repackaged with recompilation
6891  *) Updated spark-submit commandline: /home/shrinivaasanka/spark-2.1.0-bin-hadoop2.7/bin/spark-submit --class SparkGen
6892  *) Updated CLASSPATH: export CLASSPATH=/home/shrinivaasanka/spark-2.1.0-bin-hadoop2.7/jars/scala-reflect-2.11.8.jar:/
6893
6894  ----------------------------------------------------------------------------------------------------------------
6895  380. (THEORY) Machine Translation, Recursive Gloss Overlap Graphs in different Natural Languages and Graph Homomorphi
6896  - related to 178
6897  ----------------------------------------------------------------------------------------------------------------
6898  Machine Translation is equivalent to graph homomorphism between 2 recursive gloss overlap graphs of a text in 2 natur
6899  homomorphism f for translation is defined as:
6900         if (u,v) is in E(G1) then (f(u),f(v)) is in E(G2)
6901  for recursive gloss overlap graphs G1 and G2 for languages L1 (e.g. English) and L2 (e.g Tamizh) constructed from res
6902  In previous definition, u and v are two words in natural language L1 and f(u) and f(v) are two words in natural langu
6903  This relaxes the isomorphism requirement mentioned in 178. Set of all digraphs G such that there is a homomorphism fr
6904  denoted by L(H). Guessing the set of digraphs G for a digraph H implies finding set of all translations for a text fr
6905  languages to another natural language. This guessing problem is known by Dichotomy Conjecture to be either in P or NP
6906
6907  ----------------------------------------------------------------------------------------------------------------
6908  381. (FEATURE-DONE) Approximate SAT solver update - 14 February 2017
6909  ----------------------------------------------------------------------------------------------------------------
6910  *) Changed number of variables and clauses to 14 and 14 with some additional debug statements.
6911  *) After 4000+ iterations ratio of random 3SAT instances satisfied is: ~70%
6912
6913  ----------------------------------------------------------------------------------------------------------------
6914  382. (FEATURE-DONE) Java Spark Streaming with NetCat WebServer update and some analysis on HiveServer2+Spark Integrat
6915  ----------------------------------------------------------------------------------------------------------------
6916  *) Rewrote SparkGenericStreaming.java mapreduce functions with Spark 2.1.0 + Java 8 lambda functions
6917  *) Enabled Netcat Socket streaming boolean flag instead of URL socket with Jsoup
6918  *) With above changes (and no HiveServer2), socket streaming receiver works and Hive saveAsTable()/Parquet file savin
6919  *) HiveServer2 was enabled in Spark Conf by adding hive-site.xml in spark/conf directory. (Two example hive-site.xml(
6920  committed to repository.)
6921  *) Spark 2.1.0 was started with hive-site.xml to connect to HiveServer2 2.1.1. This resulted in 2 out of heapspace er
6922  in testlogs/ with exception "Error in instantiating HiveSessionState".
6923  *) Such OOM errors with HiveServer2 and Spark have been reported earlier in Apache JIRA. Instead of TTransport, Frame
6924  tried which caused "Frame size exceeds maximum frame size" errors. SASL was also disabled in hive-site.xml.
6925  *) Reason for such OOM errors in HiveServer2 seems to be heavy memory footprint of HiveServer2 and Spark together and
6926  for both cores. Might require a high-end server with large RAM size so that heapspace is set to atleast 8GB. HiveServ
6927  memory intensive application. Also Spark only supports Hive 1.2.1 and not Hive 2.1.1 which could be the reason.
6928  *) Increasing Java Heap space in spark config file (spark.executor.memory and spark.driver.memory) also did not have
6929  *) Setting HADOOP_HEAPSIZE also did not have effect. OOM error occurs while instantiating HiveSessionState and bootst
6930  (get_all_databases). Isolated beeline CLI connection works though with HiveServer2.
6931  *) Hence presently Spark Streaming works with Spark provided metastore_db Hive Support and Parquet file support.
6932  *) testlogs/ have stream processing logs for NetCat (NC) webserver.
6933
6934  ----------------------------------------------------------------------------------------------------------------
6935  383. (FEATURE-DONE) Spark Version Upgrade for Recursive Gloss Overlap Graph Intrinsic merit and Discrete Hyperbolic F
6936  Implementations - 21 February 2017
6937  ----------------------------------------------------------------------------------------------------------------
6938  *) Re-executed Interview Algorithm and Factorization Spark Cloud implementations with Spark 2.1.0 and Cython optimiza
6939  Gloss Overlap Graph construction
6940  *) Earlier these were benchmarked with Spark 1.5.x
6941  *) Added debug statements in python-src/DiscreteHyperbolicFactorizationUpperbound_Bitonic_Spark.py and file locations
6942  committed to python-src/testlogs and python-src/InterviewAlgorithm/testlogs
```

```
6943
6944  ------------------------------------------------------------------------------------------------------------------------
6945  384. (THEORY) Recursive Gloss Overlap Graphs, Intrinsic Merit and Korner Graph Entropy - related to 349 - 22 February
6946  ------------------------------------------------------------------------------------------------------------------------
6947  Graph representation of text and deriving a quantitative intrinsic merit from recursive gloss overlap definition grap
6948  WordNet onto the document has been described previously. Spectral Graph Theory qualitatively analyzes the graph in te
6949  its Laplacian. Quantitative Graph Theory provides various statistical tools to quantify a graph in terms of its compl
6950  meaningfulness of a text has been translated to the problem finding easy paths between concepts and keywords in the t
6951  graph G as follows:
6952                KE(G) = minimum(I(X,Y))
6953  where X is a randomly chosen vertex in V(G) and Y is an independent set of G (set of vertices with no edges between t
6954  probabilistic mutual information of two random variable X and Y defined as:
6955                I(X,Y) = double_summation(p(x,y)log(p(x,y)/p(x)p(y)))
6956  where p(x) is probability of X=x, p(y) is probability of Y=y and p(x,y) is a joint probability distribution of X=x an
6957
6958  An alternative definition of Korner Entropy of a graph G is :
6959                KE(G) = minimum [- sum_v_in_V(G) {1/|V(G)| * log(Pr[v in Y])}]
6960  i.e minimum of summations over product of probability of a randomly chosen vertex v in V(G) and log likelihood probab
6961
6962  Mutual information is the generalization of Shannon information entropy when two dependent random variables are invol
6963
6964  References:
6965  -----------
6966  384.1 Korner Entropy - [JaikumarRadhaKrishnan] - http://www.tcs.tifr.res.in/~jaikumar/Papers/EntropyAndCounting.pdf
6967  384.2 Entropy of Graphs - [MowshowitzDehmer] - www.mdpi.com/1099-4300/14/3/559/pdf
6968
6969  ------------------------------------------------------------------------------------------------------------------------
6970  385. (THEORY) Neural Tensor Networks Reasoning and Recursive Gloss Overlap merit scoring - 23 February 2017
6971  ------------------------------------------------------------------------------------------------------------------------
6972  Neural Tensor Networks described in 385.1 generalizes a linear neuron into a multidimensional tensor and assigns a sc
6973  between two entities e1 and e2. This is described in the context of an ontology (e.g WordNet) where a relation betwee
6974  is assigned a score based on various distance measures. Recursive Gloss Overlap graph which is a gloss definition gra
6975  between any two words - w1 "is in dictionary/gloss definition of" w2. Present implementation does not quantify the ex
6976  between two words connected by an edge in Recursive Gloss Overlap graph. This definition graph can be considered as a
6977  and each edge can be scored as a neural tensor relation between two word vertices. Cumulative sum total score of all
6978  quantitative intrinsic merit measure of the meaningfulness and intrinsic merit. Thus any document's definition graph
6979
6980  References:
6981  ----------
6982  385.1 Neural Tensor Networks - [SocherManningNg] - http://nlp.stanford.edu/pubs/SocherChenManningNg_NIPS2013.pdf
6983
6984  ------------------------------------------------------------------------------------------------------------------------
6985  386. (THEORY) Recursive Lambda Function Growth, Random Walks on Recursive Gloss Overlap graph and Text Comprehension
6986  ------------------------------------------------------------------------------------------------------------------------
6987  As mentioned in previous section, two word vertices x1 and x2 are connected by an edge weight determined by tensor ne
6988  of fuel shortage" is created from thesaurus/sdictionary/gloss definitions of an ontology:
6989
6990                        flight -------> tyre ------> wheel ---> runway
6991                          ^                            |
6992                          |                            V
6993                          |                         landing
6994                        fuel -----> shortage           |
6995                                                       V
6996                                                     gear
6997
6998  Example Random walks on previous graph:
6999  ---------------------------------------
7000                        flight ---> tyre ----> wheel ---> landing ---> gear
7001                        flight ----> fuel ----> shortage
7002  If Tensor Neurons are applied for each edge, this graph is a weighted directed graph where weight is determined by Te
7003
7004                        fuel ----> flight ---> tyre ----> wheel ---> landing ---> gear
7005
7006  is tensor neuron labelled with lambda functions:
7007
7008                        fuel --(requiredby)---> flight----(has)---> tyre ---(has)---> wheel ---(does) ---> landing ---
7009
7010  where lambda functions for each edge with word vertices as parameters are:
7011         f1 = requiredby(fuel, flight)
7012         f2 = has(flight, tyre)
7013         f3 = has(tyre, wheel)
7014         f4 = does(wheel, landing)
7015         f5 = requires(landing, gear)
7016
7017  Composition is done left associative creating following tree recursively:
7018         f5(landing,gear)
7019         f4(wheel, f5(landing, gear))
7020         f3(tyre, f4(wheel, f5(landing, gear)))
7021         f2(flight, f3(has, f4(wheel, f5(landing, gear))))
7022         f1(fuel, f2(flight, f3(tyre, f4(wheel, f5(landing, gear)))))
7023
```

```
Previous composition tree is done for each Markov Chain Random Walk on the Definition Graph. Difference between compo

Advantage of Random Walks is it mimicks the randomness of deep learning in human text comprehension. Typical visual c

References:
-----------
386.1 Random Walks Survey - http://www.cs.elte.hu/~lovasz/erdos.pdf
386.2 Random Walks on Weighted Directed Graphs - http://www.cs.yale.edu/homes/spielman/eigs/lect7.pdf

------------------------------------------------------------------------------------------------------------
387. (FEATURE-DONE) Java Spark Generic Streaming and Streaming Abstract Generator Integration - 27 February 2017
------------------------------------------------------------------------------------------------------------
*) Enabled URLSocket in Spark Generic Streaming and crawled an example facebook and twitter
streaming data stored into Parquet files and Spark metastore
*) Added Spark Streaming Data Source and Parquet Data Storage to Streaming_AbstractGenerator.py
and updated iterator to read DataFrames from word.parquet Spark Streaming storage.
*) Spark DataFrames are iterated by accessing word column ordinal and yielded to any application of
interest.
*) As an example Streaming HyperLogLogCounter implementation has been updated to read from Spark
Parquet streaming data storage and cardinality is computed
*) Example Spark Generic Streaming logs with URLSockets and HyperLogLogCounter with Spark Parquet
data storage have been added to testlogs in python-src and java-src/bigdata_analytics/spark_streaming/testlogs
*) Thus an end-to-end Java+Spark+Python streaming ETSL pipeline has been implemented where:
                - Extract is done by JSoup crawl
                - Transform is done by Spark RDD transformations
                - Store is done by Spark Hive and Parquet storage
                - Load is done by Streaming Abstract Generator iterator/facade pattern
*) Streaming_HyperLogLogCounter.py instantiates SparkSession from pyspark.sql. Usage of SparkSession requires executi

------------------------------------------------------------------------------------------------------------
388. (FEATURE-DONE) Streaming Algorithms Implementations for Spark Streaming Parquet Data Storage - 28 February 2017
------------------------------------------------------------------------------------------------------------
*) All Streaming_<algorithm> implementations have been updated to have Spark Streaming
Parquet file storage as input generators.
*) New Streaming_SocialNetworkAnalyis.py file has been added to do sentiment analysis for
Streaming Data from Spark Parquet file storage
*) Streaming_AbstractGenerator.py has been updated to filter each Spark Parquet DataFrame to remove
grammatical connectives and extract only keywords from streamed content
*) Logs for all Streaming_<algorithm>.py and Streaming_SocialNetworkAnalysis.py executed against
Spark Streaming Parquet data have been committed to testlogs/
*) Streaming_SocialNetworkAnalysis.py presently iterates through word stream generated by
Streaming_AbstractGenerator.py and does Markov Random Fields Clique potential sentiment analysis.
*) Streaming_AbstractGenerator.py Spark Parquet __iter__() clause presently returns word stream which
can be optionally made to return sentences or phrases of specific length
*) All python streaming implementations instantiating Streaming_AbstractGenerator.py have to be executed with $SPARK_

------------------------------------------------------------------------------------------------------------------------
389. (FEATURE-DONE) Major rewrite of BackPropagation Implementation for arbitrary number of Neuron input variables la
------------------------------------------------------------------------------------------------------------------------
*) BackPropagation algorithm code has been rewritten for arbitrary number of input layer, hidden layer and output lay
*) Some Partial Differential Equations functions have been merged into one function.
*) This was executed with 6 variable input layer neuron, 6 variable hidden layer and 6 variable output layer with 2*6
for each of 72 activations.
*) Logs for this have been committed to testlogs/
*) For 100000 iterations, error tapers to ~10^-29 as below
------------------------------------------------------------------------------------------------------------
Error after Backpropagation- iteration : 99998
1.05827458078e-29
Layers in this iteration:
###############
Input Layer:
###############
[0.23, 0.11, 0.05, 0.046, 0.003, 0.1]
###############
Hidden middle Layer:
###############
[0.07918765738490302, 0.17639329000539292, 0.3059104560477687, 0.06819897810162112, 0.13596389556531566, 0.1809236742
###############
Output Layer:
###############
[0.29999999999999977, 0.53, 0.11000000000000039, 0.09000000000000002, 0.010000000000004578, 0.2099999999999999]
Weights updated in this iteration:
[0.048788506679748, 0.11426171324845727, 0.24603459114714846, 0.0795780558315572, 0.17784209754093078, 0.189016659729
Recomputing Neural Network after backpropagation weight update
Error after Backpropagation- iteration : 99999
1.05827458078e-29
Layers in this iteration:
###############
Input Layer:
###############
[0.23, 0.11, 0.05, 0.046, 0.003, 0.1]
```

```
###############
Hidden middle Layer:
###############
[0.07918765738490302, 0.17639329000539292, 0.3059104560477687, 0.06819897810162112, 0.13596389556531566, 0.1809236742
###############
-----------------------------------------------------------------------------------------------------------------------

-----------------------------------------------------------------------------------------------------------------------
390. (FEATURE-DONE) Convolution Network Final Neuron Layer Integrated with BackPropagation Implementation- 3 March 20
-----------------------------------------------------------------------------------------------------------------------
*) New python module has been added to repository to invoke BackPropagation algorithm
implementation in final neuron layer of convolution network.
*) This creates a multilayer perceptron from maxpooling layer variables as input layer and
backpropagates for few iterations to do weight updates.
*) Maxpooling layer is a matrix of 5*5=25 variables. With 25 variable input layer, backpropagation
is done on 25*25*2=1250 activation edges with weights(25 inputs * 25 hidden + 25 hidden * 25 outputs).
*) Logs for this have been added to testlogs/
*) Presently expected output layer has been hardcoded to some constant

-----------------------------------------------------------------------------------------------------------------------
391. (THEORY) Prime-Composite Complementation/2-coloring, Riemann Zeta Function and Sum of Eigenvalues - Elementary A
- related to 19, 24, 319, 323, 338, 371
-----------------------------------------------------------------------------------------------------------------------
Special case of complementation/2-coloring is prime-composite coloring where set of natural numbers is 2-colored as p
Here terminology of complementation and 2-coloring is used interchangeably. But complementation is a generic notion f
reals too. Riemann Zeta Function and truth of Riemann Hypothesis imply a pattern in distribution of primes and hence
coloring. Thus Riemann Zeta Function is a 2-coloring scheme for set of natural numbers.

Let q^s + q^(1-s) = v be an eigenvalue of a graph where s is a complex exponent (s=a+ib). It can be written as:
       q^2s + q = vq^s
       q^2(a+ib) + q = vq^(a+ib)
       q^2a*q^2ib + q = vq^a*q^ib
       q^2a(e^i2blogq) + q = vq^a[e^iblogq] by rewriting q^ib = e^iblogq
       q^2a[cos(2blogq) + isin(2blogq)] + q = vq^a[cos(blogq) + isin(blogq)]
Equating Re() and Im():
       q^2a[cos(2blogq)] + q = vq^a[cos(blogq)]
       q^2a[sin(2blogq)] = vq^a[sin(blogq)]
Ratio of Re() and Im() on both sides:
       q^2a[sin(2blogq)]               vq^a[sin(blogq)]
     -------------------------  =      ----------------
       q^2a[cos(2blogq)] + q           vq^a[cos(blogq)]

   q^2a[2*sin(blogq)*cos(blogq)]              [sin(blogq)]
     ------------------------  =          ----------------
       q^2a[cos(2blogq)] + q                [cos(blogq)]

       q^2a[2*cos(blogq)*cos(blogq)] =   q^2a[cos(2blogq)] + q

       q^2a[2*cos(blogq)*cos(blogq)] =   q^2a[cos(blogq)*cos(blogq) - sin(blogq)*sin(blogq)] + q

       q^2a[cos(blogq)*cos(blogq) + sin(blogq)*sin(blogq)] = q

       q^2a = q => a = 0.5

When s=0.5 with no imaginary part and q+1 is regularity of a graph, v = 2*sqrt(q) and corresponding graph is a q+1 re

[This is already derived in:
https://sites.google.com/site/kuja27/RamanujanGraphsRiemannZetaFunctionAndIharaZetaFunction.pdf,
https://sites.google.com/site/kuja27/RZFAndIZF_25October2014.pdf,
http://sourceforge.net/p/asfer/code/HEAD/tree/python-src/ComplFunction_DHF_PVsNP_Misc_Notes.pdf]

Let H = {G1,G2,G3,...} be an infinite set of graphs with eigenvalues 1^s + 1^(1-s), 2^s + 2^(1-s), 3^s + 3^(1-s),....

Sum of these eigenvalues for infinite set of graphs has following relation to Riemann Zeta Function for s=w+1:
Sum_q=1_to_inf(q^s + q^(1-s)) = [1 + 1/2^w + 1/3^w + ...] + [1 + 2^(w+1) + 3^(w+1) + ...] = RiemannZetaFunction + [1

Sum_q=1_to_inf(q^s + q^(1-s)) -  [1 + 2^(w+1) + 3^(w+1) + ...] = RiemannZetaFunction

Thus Riemann Zeta Function is written as difference of sum of eigenvalues for an infinite set of graphs and another i
Hermitian Real Symmetric matrices have real eigenvalues which is true for undirected graphs. For directed graphs, eig

References:
-----------
391.1 Complex Analysis - [Lahs Alfors] - Page 213 - Product Development, Riemann Zeta Function and Prime Numbers
391.2 Honeycombs and Sums of Hermitian Matrices - http://www.ams.org/notices/200102/fea-knutson.pdf

-----------------------------------------------------------------------------------------------------------------------
392. (FEATURE-DONE) DeepLearning (ConvolutionNetwork-BackPropagation) JPEG files with Python Imaging Library (PILlow)
-----------------------------------------------------------------------------------------------------------------------
*) Added import of PIL Image library to open any image file and convert into a matrix in Integer mode
*) This matrix is input to Convolution and BackPropagation code
```

```
*) Logs of Convolution and BackPropagation for sample images have been committed to testlogs/

------------------------------------------------------------------------------------------------------------------------
393. (FEATURE-DONE) DeepLearning (ConvolutionNetwork-BackPropagation) tobit() update - 7 March 2017
------------------------------------------------------------------------------------------------------------------------
*) Updated python-src/DeepLearning_ConvolutionNetwork_BackPropagation.py tobit() function to map each pixel to a deci
*) Increased BackPropagation iteration to 100.
*) Logs for this have been committed to testlogs/. With this final neurons for max pooling layer are sensitive to cha

------------------------------------------------------------------------------------------------------------------------
394. (FEATURE-DONE) DeepLearning (ConvolutionNetwork-BackPropagation) image_to_bitmatrix() update - 8 March 2017
------------------------------------------------------------------------------------------------------------------------
*) Refactored the tobit() function - moved it to a new directory image_pattern_mining and reimplemented it with a new
image_to_bitmatrix() function in ImageToBitMatrix.py by enumerating through the pixel rows and applying map(tobit)
*) This removed the opaqueness related 255 appearing for all pixels in bitmap.
*) Each pixel is multiplied by a fraction. Mapping it to binary digits 0,1 based on [0-127],[128-255] intervals for p
does not work well with BackPropagation of MaxPooling in Convolution Network. Instead multiples of a small fraction m
MaxPooling layer quite receptive to changes in image pixel values.
*) Added few handwriting recognition example images for numbers 1(two fonts),2 and 8.
*) Logs for this have been committed to testlogs/
----------------------------------------------------------
Following are Maxpooling Neurons
----------------------------------------------------------
Final Layer of Inference from Max Pooling Layer - BackPropagation on Max Pooling Layer Neurons
Inference from Max Pooling Layer - Image: [0.1452950826564989, 0.15731736449115244, 0.16984767785665972]
Inference from Max Pooling Layer - Image: [0.14920888863234688, 0.16193600589418844, 0.17528092024601222]
Inference from Max Pooling Layer - Image: [0.13197897244871004, 0.1417448800619895, 0.15171424525284838]
Inference from Max Pooling Layer - Image: [0.14007345008310818, 0.15118749472569865, 0.16267822705412283]
Inference from Max Pooling Layer - Image: [0.07847441088145349, 0.0807616625716103, 0.08303607840796547]
Inference from Max Pooling Layer - Image: [0.08983732685398557, 0.09365789540391084, 0.09744765711798722]
Inference from Max Pooling Layer - Image: [0.0903328782140764, 0.09421672093823663, 0.09806798136073994]
Inference from Max Pooling Layer - Image: [0.08761036574911765, 0.09113121170496671, 0.09462405037074482]
('Inference from Max Pooling Layer - Example 11:', [0.1619896774094416, 0.1615169723530323, 0.15009965578039716])
('Inference from Max Pooling Layer - Example 12:', [0.14303879829783814, 0.12068614560924568, 0.07211705852669059])
('Inference from Max Pooling Layer - Example 21:', [0.1932801269861012, 0.11587916253160746, -0.007889615456070018])
('Inference from Max Pooling Layer - Example 22:', [0.17605214289097237, 0.19326380725531284, 0.2114152778184319])
('Inference from Max Pooling Layer - Example 3:', [0.062097695970929206, 0.062097695970929206, 0.062097695970929206])
('Inference from Max Pooling Layer - Example 41:', [0.2976324086103368, 0.32891470561393965, 0.34986397693929844])
('Inference from Max Pooling Layer - Example 42:', [0.22669904395788065, 0.26336097548290116, 0.3065443950445386])
('Inference from Max Pooling Layer - Example 51:', [0.11935155170571034, 0.12477637373255022, 0.12911872514904305])
('Inference from Max Pooling Layer - Example 52:', [0.12675869649313706, 0.12051203261825084, 0.10578392639393523])
----------------------------------------------------------
Previous logs are grep-ed from testlogs/. Handwritten Numeral recognitions (with high noise) are done in following Ma
[Number 1 - Font 1] Inference from Max Pooling Layer - Image: [0.07847441088145349, 0.0807616625716103, 0.0830360784(
[Number 1 - Font 2] Inference from Max Pooling Layer - Image: [0.08983732685398557, 0.09365789540391084, 0.0974476571
[Number 2] Inference from Max Pooling Layer - Image: [0.0903328782140764, 0.09421672093823663, 0.09806798136073994]
[Number 8] Inference from Max Pooling Layer - Image: [0.08761036574911765, 0.09113121170496671, 0.09462405037074482]

Contrasting this with bitmap numerals inscribed in 2 dimensional arrays (with low noise), similar elements have close
[Number 0 - Font 1] ('Inference from Max Pooling Layer - Example 11:', [0.1619896774094416, 0.1615169723530323, 0.150
[Number 0 - Font 2] ('Inference from Max Pooling Layer - Example 12:', [0.14303879829783814, 0.12068614560924568, 0.0
[Number 8 - Font 1] ('Inference from Max Pooling Layer - Example 21:', [0.1932801269861012, 0.11587916253160746, -0.0
[Number 8 - Font 2] ('Inference from Max Pooling Layer - Example 22:', [0.17605214289097237, 0.19326380725531284, 0.2
[Null pattern] ('Inference from Max Pooling Layer - Example 3:', [0.062097695970929206, 0.062097695970929206, 0.06209
[Letter X - Font 1] [('Inference from Max Pooling Layer - Example 41:', [0.2976324086103368, 0.32891470561393965, 0.3
[Letter X - Font 2] ('Inference from Max Pooling Layer - Example 42:', [0.22669904395788065, 0.26336097548290116, 0.3
[Number 1 - Font 1] ('Inference from Max Pooling Layer - Example 51:', [0.11935155170571034, 0.12477637373255022, 0.1
[Number 1 - Font 2] ('Inference from Max Pooling Layer - Example 52:', [0.12675869649313706, 0.12051203261825084, 0.1

------------------------------------------------------------------------------------------------------------------------
395. (THEORY) Coloring Real Numbers and Complement Functions - 8 March 2017 - related to 323, 338
------------------------------------------------------------------------------------------------------------------------
So far equivalence of 2-coloring schemes and integer valued complement functions have been mentioned. Most generic pr
is to find coloring schemes (or) functions which "complement" real and complex n-dimensional surfaces(i.e sets of val
its complement g create a disjoint set cover of the real/complex planes). Ramsey theory pertains to coloring integer
the problem of coloring the real line as minimum number of colors required to color the real line such that no two po
graph where two vertices a and b are adjacent iff distance between a and b on real line is in distance set. This is s
of coloring compared to complementation and complement graphs representation mentioned in 338. For example, following
       f(0) = [1.0,1.11]
              g(0) = [1.12,1.23]
       f(1) = [1.24,1.30]
              g(1) = [1.31,1.68]
       f(2) = [1.69,2.0]
has colored regions of length [0.11,0.11,0.06,0.37,0.31] and can be 2-colored with f and g. If the distance set is {0
points 1.32 and 1.33 are 0.01 apart but have same color (belong to same function g). Similarly points 1.69 and 1.71 a
have same color f. Thus complementation over reals relaxes the coloring conditions significantly. Coloring real line
omplementation with distance set requirement while Coloring integers is equivalent to complementation. Rather, distan
complementation is of size 1 containing length of largest monochromatic streak - in previous example it is {0.37}.

References:
----------
```

```
395.1 Coloring Real Line - [EggletonErdosSkilton] - http://ac.els-cdn.com/0095895685900395/1-s2.0-0095895685900395-ma
395.2 Coloring Reals - http://webbuild.knu.ac.kr/~trj/HN.pdf

------------------------------------------------------------------------------------------------------------------------
396. (FEATURE-DONE) Deep Learning Recurrent Neural Network Gated Recurrent Unit (RNN GRU) Implementation - 9 March 20
------------------------------------------------------------------------------------------------------------------------
*) This commit implements Gated Recurrent Unit algorithm (most recent advance in deep learning - published in 2014) w
simplification of RNN LSTM by reducing number of gates (input, cell, forget, output are mapped to cell, reset, update
*) Logs for this have been uploaded to testlogs/ showing convergence of gates and state.

------------------------------------------------------------------------------------------------------------------------
397. (FEATURE-DONE) Recursive Lambda Function Growth + Tensor Neuron (NTN) Intrinsic Merit - 13 March 2017 - related
------------------------------------------------------------------------------------------------------------------------
*) This commit rewrites the Recursive Lambda Function Growth implementation by adding a new lambda function growth
function.
*) This new function creates a definition graph with Recursive Gloss Overlap algorithm from a text and computes all p
*) For each such shortest path edges, an AVL lambda function composition balanced tree is grown (i.e every path is a
*) Tensor Neuron Lambda Function is computed for each subtree root by following relation evaluated on a stack:
                function(operand1, operand2)
where operand1, operand2 and function are successively popped from AVL composition tree stack representation
*) Tensor Neuron triplet (operand1, function, operand2) is evaluated presently as maximum Wu-Palmer similarity of Syn
*) Sum of tensor neuron weights for all random walks composition tree evaluation is printed as merit of the text.
*) Logs for this have been committed to testlogs

------------------------------------------------------------------------------------------------------------------------
398. (FEATURE-DONE) Korner Entropy Intrinsic Merit for Recursive Gloss Overlap graphs - 14 March 2017 - related to 38
------------------------------------------------------------------------------------------------------------------------
*) Implemented  Korner Entropy intrinsic merit for Recursive Gloss Overlap graph.
*) function korner_entropy() finds maximal independent sets for all vertices and chooses the minimum entropy such tha
*) Logs for this have been committed to testlogs/

Previous two intrinsic merit measures - tensor neuron network and korner entropy - sufficiently quantify the complexi

------------------------------------------------------------------------------------------------------------------------
399. (FEATURE-DONE) Software Analytics update - 15 March 2017
------------------------------------------------------------------------------------------------------------------------
*) Software Analytics Deep Learning code has been updated to include all Deep Learning implementations: RNN LSTM, RNN
and ConvolutionNetwork+BackPropagation
*) logs for this have been committed to software_analytics/testlogs/

------------------------------------------------------------------------------------------------------------------------
400. (USECASE) NeuronRain Usecases - Software (Log) Analytics and VIRGO kernel_analytics config - 16 March 2017 and 2
------------------------------------------------------------------------------------------------------------------------
Software Analytics Deep Learning implementation in python-src/software_analytics/ sources its input variables - CPU%,

       If the CPU% is > 75% and Memory% > 75% and TimeDuration% > 75% then
               do a kernelwide overload signal
       else
               do nothing

Sigmoid function for this usecase is:
       sigma(4/9 * x1 + 4/9 * x2 + 4/9 * x3) where x1 = CPU%, x2 = Memory% and x3 = TimeDuration%
which is 1 when all variables reach 75%.

Spark Streaming Log Analyzer ETLs htop data to .parquet files and Software Analytics code reads it, deep-learns above

An inverse of this usecase is to find the weights for x1,x2,x3 when system crashes. This requires a gradient descent

------------------------------------------------------------------------------------------------------------------------
401. (THEORY) Mechanism Design, Condorcet Jury Theorem, Derandomization, Gibbard-Satterthwaite Theorem and Designing
- 27 March 2017 , 30 March 2017 ,31 March 2017, 18 April 2017 - related to 369 and all other P(Good) circuit related
------------------------------------------------------------------------------------------------------------------------
Gibbard-Satterthwaite theorem prevents any Social Choice Function defined on atleast 3 elements from being non-dictat

A simple example of strategic/tactical voting and Gibbard-Satterthwaite theorem is given in http://rangevoting.org/Ir

This implies for more than 3 candidates, RHS of P(Good) series for majority voting can never converge because there i

Condorcet Jury Theorem mentioned in references below, is exactly the P(Good) binomial series summation and has been s

Corollary: From 129 and 369, depth of Majority voting circuit composition can be unbounded and if all voter social ch

If Majority Voting Circuit is in BPP, which is quite possible if number of voters is polynomial in number of variable

Locality Sensitive Hashing accumulates similar voters in a bucket chain. The social choice function is the distance f
       *) Locality Sensitive Hashing (LSH) for clustering similar voters who vote for same candidate
       *) Sorting the LSH and find the maximum size cluster (bucket chain)

References:
-----------
401.1 Mechanism Design, Convicting Innocent, Arrow and GibbardSatterthwaite Theorems - https://ocw.mit.edu/courses/el
```

401.2 Electoral Fraud, Gibbard-Satterthwaite Theorem and its recent extensions on manipulative and strategic voting,
401.3 Most important point of this document (which I was searching for 11 years) - Condorcet Jury Theorem in Politica
"... Condorcet's Jury theorem applies to the following hypothetical situation: suppose that there is some
401.4 Strategic Voting and Mechanism Design - [VinayakTripathi] - https://www.princeton.edu/~smorris/pdfs/PhD/Tripath
401.5 Condorcet Jury Theorem and OCR - Section 2 - http://math.unipa.it/~grim/Jlamlouisa.PDF
401.6 Condorcet Jury Theorem graph plots - http://www.statisticalconsultants.co.nz/blog/condorcets-jury-theorem.html
401.7 Probabilistic Aspects of Voting - Course Notes - www.maths.bath.ac.uk/~ak257/talks/Uugnaa.pdf - Condorcet Jury
401.8 Law of Large Numbers and Weighted Majority - [Olle Haggstrom, Gil Kalai, Elchanan Mossel] - https://arxiv.org/a
401.9 Thirteen Theorems in search of truth - http://www.socsci.uci.edu/~bgrofman/69%20Grofman-Owen-Feld-13%20theorems
401.10 Margulis-Russo Formula and Condorcet Jury Theorem - http://www.cs.tau.ac.il/~amnon/Classes/2016-PRG/Analysis-C
401.11 Percolation, Sharp Threshold in Majority, Condorcet Jury Theorem and later results, Influence, Pivotality - ht
401.12 Complexity Inclusion Graph - https://www.math.ucdavis.edu/~greg/zoology/diagram.xml - BPEXP=EXP implies a sigr

---
402. (FEATURE-DONE) LSH Index implementation for NeuronRain AsFer Text and Crawled Web Documents Intrinsic Merit Rank
---
*) This commit adds a Locality Sensitive Hashing based inverted index for scrapy crawled **HTML** website pages.
*) It is a derivative of LSH based similarity clustering implemented in python-src/LocalitySensitiveHashing.py
*) It has to be mentioned here that ThoughtNet is also an inverted index for documents with additional features (i.e
classifier based than raw string similarity)
*) LSHIndex uses Redis Distributed Key Value Persistent Store as hashtable backend to store index.
*) LSHIndex is not string to text inverted index, but hashed string to text inverted index (something like a digital
*) Presently two types of hashes exist: primitive ordinal and MD5 hash
*) Only one hash table is implemented on Redis with multiple hash functions with random choice.
*) Presently 50 hash functions with 50 entries hash index has been implemented.

---
403. (FEATURE-DONE) ThoughNet Index implementation for NeuronRain AsFer Text and Crawled Web Documents Intrinsic Meri
---
*) This commit implements an inverted index on top of ThoughtNet
*) ThoughtNet is a hypergraph stored in file and on Neo4j Graph Database
*) JSON loads ThoughtNet edges and hypergraph files and queries by classes returned by Recursive Gloss Overlap classi
*) Querying Neo4j is also better, but file representation of ThoughtNet is quite succinct and scalable because it jus
encodes each edge (= a web crawled **HTML** document)
*) file representation of ThoughtNet views the hypergraph as stack vertices interconnected by document id(s) similar
versioning system. Having similar view on a graph database doesn't look straightforward

---
404. (FEATURE-DONE) Initial commits for Indexless Hyperball Recursive Web Crawler - 7 April 2017
---
*) An indexless crawler which tries to find a url matching a query string in a recursive crawl creating a hyperball c
*) This is similar to Stanley Milgram, Kleinberg Lattice Small World Experiments and recent Hyperball algorithm for F
concluded that Facebook has 3.74 degrees of freedom.
*) Starts with a random url on world wide web.
*) Success probability depends on average number of hops over large number of queries.
*) This is a Randomized Monte Carlo Crawler and does a Depth First Search from an epicentre pivot url to start with
*) Hyperball References:
1) http://webgraph.di.unimi.it/docs/it/unimi/dsi/webgraph/algo/HyperBall.html
2) Facebook Hyperball - https://arxiv.org/pdf/1308.2144.pdf

---
405. (THEORY) Indexless Web Search, World Wide Web Graph Property Testing, On-demand Indexing - related to 404 - 9 Ap
---
Almost all known web search engines rely on a pre-built index created from already crawled WWW graph. Is a pre-built
web search? Can the small world property of world wide web graph (i.e property of graph such that easy short paths ex
of vertices as proved in Stanley Milgram experiment, Kleinberg Lattice), be applied to find a url matching a search c
vertex? Facebook graph has almost a billion user vertices and on an average any person is connected to any other with
Can this be generalized to world wide web? Traditional book index maps word queries to pages having the words. If ent
into a huge definition graph by Recursive Gloss Overlap graph, then instead of looking-up index, starting at a random
word should be easily reachable if graph has small world property. This realworld problem is already studied as Graph

References:

405.1 Small World Property - [WattsStrogatz] -http://www.nature.com/nature/journal/v393/n6684/full/393440a0.html

---
406. (FEATURE-DONE) Commit - NeuronRain AsFer-CPython Extensions VIRGO64 system call invocations - 20 April 2017
---
*) VIRGO64 system calls are invoked from Python code by CPython extensions.
*) Separate folder cpython_extensions64/ has been created for 64bit VIRGO kernel
*) Requires Python 64-bit version
*) Logs for VIRGO64 memory and filesystem systemcalls-to-drivers have been committed to testlogs
*) With this Complete Application Layer-SystemCalls-Drivers request routing for python applications deployed on VIRGO

---
407. (FEATURE-DONE) Commits - NeuronRain AsFer Boost C++-Python - VIRGO64 System calls + Drivers invocation - 26 Apri
---
*) VIRGO64 system calls are invoked from C++ by Boost::Python extensions
*) Kernel Logs for VIRGO KMemCache, Clone and FileSystem Calls Boost::python invocations have been committed to testl
*) Boost version used is 1.64.0
*) setup.py has been updated with library_dirs config variable

*) virgofstest.txt for filesystem calls has been committed to testlogs/

---

408. (FEATURE-DONE) Commits - 28 April 2017 - NeuronRain AsFer-Boost::Python-C++ - VIRGO64 systemcalls and drivers in

---

*) VIRGO64 clone, kmemcache and filesystem system calls were invoked from Python-C++ boost extensions again and reprc
without kernel panics.
*) Logs, persisted disk file written by filesystem system calls and rebuilt boost-python C++ extensions have been com

---

409. (THEORY) Condorcet Jury Theorem, Collaborative Filtering, Epistemological Democracy, Network Voting in WWW Link
Majority Function and Correctness of Majority Vote Ranking - 16 May 2017, 23 May 2017

Let v1, v2 be two vertices being good and bad choices respectively. In link graph all incoming edges to a vertex v1 a
votes for that vertex. From Condorcet Jury Theorem if all incoming links occur with probability p > 0.5, then probabi

More generically, vertex vx receives votes from vertices vy1,vy2,vy3,...,vyn with decision correctness probability p.

Collaborative Filtering in Recommender Systems is the most generalized way of Majority Voting where a matrix of users

References:

409.1 Condorcet Jury Theorem and the truth on the web - http://voxpublica.no/2017/03/condorcets-jury-theorem-and-the-
409.2 Webometrics, Goodness of PageRank and Condorcet Jury Theorem - [MastertonOlssonAngere] - https://link.springer.
409.3 Generalized Condorcet Jury Theorem, Free Speech, Correlated Voting (dependence of voters) - https://www.jstor.o

---

410. (FEATURE-DONE) Commits - 17 May 2017 - NeuronRain AsFer-VIRGO64 Boost-C++-Python invocations

---

*) Boost Python C++ VIRGO64 system calls were tested repeatedly in a loop
*) All three system call subsystems - virgo_clone, virgo_kmemcache, virgo_filesystem - work well without any kernel p
*) But a strange thing was observed: virgo filesystem system calls were not appending to disk file but when run as
"strace -f python asferpythonextensions.py" disk file is written to (a coincidence or strace doing something special)

---

411. (THEORY) Objective and Subjective Value Judgement, Text Ranking and Condorcet Jury Theorem - 29 May 2017 and 4 J
- Related to 202, 385, 386 and all Recursive Lambda Function Growth and Recursive Gloss Overlap algorithms sections i

---

Objective merit or Intrinsic merit of a document is the measure of meaningfulness or information contained in a docum

Following are few real-world examples in addition to academic credentials example in 202:
*) Soccer player, Cricket player or a Tennis player is measured intrinsically by number of goals scored, number of ru
*) Merits of students are judged by examinations (question-answering) and not by majority voting by faculty. Thus que

Present ranking algorithms are mostly majority voting (perception) oriented which is just half truth. Other half is t

But how to measure "number of goals etc.," of a document vis-a-vis the rest? This is where Computational Linguistic T

Condorcet Jury Theorem formalizes above intuition and bridges two worlds - objective reality and subjective perceptic

Crucial Observation is: Merit creates Centrality (Social prestige) and not the opposite - Prestige can not create mer

Alternatively, interview of document is simply the rank of the document in terms of intrinsic graph complexity merit
*) Ranking text document by intrinsic graph complexity/entropy merit
*) Relevance of the text to a query - graph edit distance

Here again another question can be raised: Why is a graph representation of text an apt objective intrinsic merit mea

There is special case of intrinsic merit: For example, reading a story creates a mental picture of it as a graph of e

References:

411.1 The Meaning of Meaning - http://courses.media.mit.edu/2004spring/mas966/Ogden%20Richards%201923.pdf
411.2 WordNet and Word2Vec - https://yaledatascience.github.io/2017/03/17/nnnlp.html - In word2vec words from text ar
411.3 Random Walks on WordNet - http://anthology.aclweb.org/N/N15/N15-1165.pdf - Recursive Lambda Function Growth alg

---

412. (FEATURE-DONE) AngularJS - Tornado GUI-REST WebService - Commits - 1 June 2017

NeuronRain AngularJS RESTful client for Tornado webserver and others

---

*) AngularJS support has been added to NeuronRain GUI-WebServer with a new webserver_rest_ui/NeuronRain_AngularJS_RES
and renders angularjs Model-View-Controller templates
*) New AngularJS Model, View and Controller script-html templates have been added to angularjs directory
*) LSHIndex.py has been updated to have commandline arguments for index queries
*) Hyperball Crawler pivot epicentre url has been changed and an example query "Chennai" was found to be matching wit
*) NeuronRain_REST_WebServer.py has been updated to print the logs for the script execution to browser console with s

---

413. (THEORY) Graph Neural Networks, Tensor Neurons and Recursive Lambda Function Growth Intrinsic Merit - 4 June 201

```
8 June 2017 - related to 411
----------------------------------------------------------------------------------------------------------------------
Definition Recursive Gloss Overlap Graph with word-word edges as Neuron Tensor Network relations having potentials wa
        p(f(w1,w2)) = p(w1) * p(w2) for some operator *

Rather than mere summation of potentials of edges following minmax criterion extracts the most meaningful random walk
*) Minimum Neuron Tensor Potential edge of each random walk is found - path minimum potential
*) Maximum of all minimum path potentials extracts a random walk lambda composition tree which is the most probable i

Event Related Potentials (ERP) mentioned in 202 are unusual spikes in EEG of brain (N400 dataset) when unrelated word

Graph Neural Networks are recent models of neural network which generalize to a graph. In a Graph Neural Network, pot

Data on websites can be classified into 3 categories: 1) text 2) voice 3) images and videos. Intrinsic merit which is

Ranking text by Graph Tensor Neuron Network intrinsic merit potential can be done in multiple ways:
- Rank Vertices and Edges by potential
- Rank the random walk lambda function composition tree by potential
- Rank by Korner Entropy of the Graph Tensor Neuron Network and other qualitative and quantitative graph complexity m
and so on.

Example lambda composition tree on a random walk of recursive gloss overlap graph:
       f1 = requiredby(fuel, flight)
       f2 = has(flight, tyre)
       f3 = has(tyre, wheel)
       f4 = does(wheel, landing)
       f5 = requires(landing, gear)
       f1(fuel, f2(flight, f3(tyre, f4(wheel, f5(landing, gear)))))

Previous example random walk lambda composition tree has been constructed in 385 and 386. Potentials for Tensor Neuro

Difference in Graph Tensor Neuron Network mapping of a text is: Each lambda composition tree is evaluated as a Graph

Intuition for Graph Tensor Neuron Network is below:
*) Random walk on recursive gloss overlap graph simulates randomness in cognitive cerebral text comprehension.
*) Tensor Neuron relatedness potential between two word vertices in definition graph quantifies relevance and meaning
*) Lambda composition tree-graph neural network for each random walk on definition graph simulates how meaning is rec

References:
-----------
413.1 Graph Neural Networks -  http://repository.hkbu.edu.hk/cgi/viewcontent.cgi?article=1000&context=vprd_ja

----------------------------------------------------------------------------------------------------------------------
414. (FEATURE-DONE) Recursive Lambda Function Growth - Graph Tensor Neuron Network Implementation - Commits - 9 June
----------------------------------------------------------------------------------------------------------------------
(*) Recursive Lambda Function Growth implementation has been updated to compute Graph Tensor Neuron Network intrinsic
(*) Each lambda function composition tree of a random walk on recursive gloss overlap graph is evaluated as a Graph N
(*) Tensor neuron potential for relation edges in Graph Neural Network has been hardcoded at present and requires a d
connective relations similar to EEG dataset for brain spikes in electric potentials.

----------------------------------------------------------------------------------------------------------------------
415. (FEATURE-DONE) Graph Tensor Neuron Network - implementation update - Commits - 10 June 2017
----------------------------------------------------------------------------------------------------------------------
*) Changed the subtree graph tensor neuron network computation

*) Children of each subtree are the Tensor Neuron inputs to the subtree root
Each subtree is evaluated as a graph neural network with weights for
each neural input to the subtree root.

*) WordNet similarity is computed between each child and subtree root and is presently assumed as Tensor Neuron
relation potential for the lack of better metric to measure word-word EEG potential.
If a dataset for tensor neuron potential is available, it has to to be looked-up and numeric
potential has to be returned from here.

*) Finally a neuron activation function (simple 1-dimensional tensor) is computed and returned to the subtree root fo

*) logs for the this have been committed to testlogs. Presently graph tensor neuron network intrinsic merit is a very
because of the decimal values of similarity and tensor neurons, but quite receptive to small changes.

----------------------------------------------------------------------------------------------------------------------
416. (FEATURE-DONE) Script for Querying Index (LSH and ThoughtNet) and Ranking the results with Recursive Lambda Func
- Commits - 13 June 2017
----------------------------------------------------------------------------------------------------------------------
*) New python script QueryIndexAndRank.py has been committed for retrieving results from Index matching a query and r
*) It queries both Locality Sensitive Hashing and ThoughtNet indices
*) Functions of classes have been parametrized for invocation across modules
*) Both LSH and ThoughtNet indices have been recreated
*) Ranking is done by Recursive Lambda Function Growth algorithm which is a superset of Recursive Gloss Overlap
*) This script was tested via NeuronRain Tornado RESTful GUI

----------------------------------------------------------------------------------------------------------------------
```

```
417. (FEATURE-DONE) Graph Tensor Neuron Network Intrinsic Merit and QueryIndexAndRank update - Commits - 15 June 2017
----------------------------------------------------------------------------------------------------------------------
*) Graph Tensor Neuron Network Intrinsic Merit computation has been changed - Intrinsic merits of all random walks ar
*) Some bugs fixed and debug prints have been changed
*) logs have been committed to testlogs/

----------------------------------------------------------------------------------------------------------------------
418. (FEATURE-DONE) Hyperball crawler update - Commits - 19 June 2017
----------------------------------------------------------------------------------------------------------------------
Integrated Recursive Lambda Function Growth Intrinsic Merit Score(Graph Tensor Neuron Network and Korner Entropy)  in

----------------------------------------------------------------------------------------------------------------------
419. (THEORY) Thought Experiment of Generic Intrinsic Merit and Condorcet Jury Theorem - 21 June 2017
----------------------------------------------------------------------------------------------------------------------
Intrinsic merit so far is restricted to text documents. Subjective ranking measures are like mirrors which reflect th
candidate. Each vote in the majority voting is similar to an image of the votee projected on to the voter mirror. Som
while others don't. Efficiency of a voter mirror is equivalent to decision correctness of a voter. Present web rankin
(*) Peruse academic records and translate to score
(*) Peruse work records and translate to score
(*) Translate awards received to score
(*) Translate brain imageing data to score (EEG and fMRI)
(*) Translate IQ or EQ to score
Above is not a formal algorithm but tries to intuit how it may look like. Human Resource Analytics are done as above

Objective Intrinsic Merit = Vote by an algorithm
Subjective Ranking = Votes by people translated into rank by an algorithm (present ranking algorithms rely on incomin

Thus Generic Intrinsic Merit removes human element completely while assessing merit (closer to AI).Goodness when it a

(*)Question: Why should intrinsic merit be judged only in this way?
(*)Answer: This is not the only possible objective intrinsic merit judgement. There could be other ways too. Disclaim

(*)Question: Wouldn't cerebral representation vary from person to person and thus be subjective?
(*)Answer: Yes, but there are standardized event related potential datasets gathered from multiple neuroscience exper

(*)Question: Isn't perception based ranking enough? Why is such an intrusive objective merit required?
(*)Answer: Yes and No. Perception majority voting based ranking is accurate only if  all voters have decision correct

It was mentioned earlier that Fourier analysis of visuals and voice is the measure of intrinsic merit. Following defi
Generic intrinsic merit of an entity - text, visual or voice - is the complexity of cerebral representation potential

References:
-----------
419.1 Event Related Potentials for attractive facial recognition - https://labs.la.utexas.edu/langloislab/research/ev
419.2 Event Related Potentials - http://cognitrn.psych.indiana.edu/busey/eegseminar/pdfs/Event-Related%2520Potentials

----------------------------------------------------------------------------------------------------------------------
420. (THEORY) Goodness of Link Graph Majority Voting and Complex Plane representation of merit - 22 June 2017 and 23
----------------------------------------------------------------------------------------------------------------------
Following is an example link graph in world wide web with weights for each edge:
       y1 - x1 : 0.25
       y1 - x2 : 0.25
       y1 - x3 : 0.25
       y1 - x4 : 0.25
implying y1 votes to x1,x2,x3,x4 with weight 0.25 each.

Following is the corresponding initial decision correctness (goodness) graph with goodness for each edge:
       y1 - x1 : 0.25
       y1 - x2 : 0.25
       y1 - x3 : 0.375
       y1 - x4 : 0.125
implying y1 votes to x1 with correctness 0.25, x2 with correctness 0.25, x3 with correctness 0.375 and x4 with correc

Markov iteration on the decision correctness graph (similar to random walk on link graph) tends to a stationary distr
       summation(weight(edge) * goodness(edge))
For previous example cumulative goodness of voter y1 is:
       0.25*0.25 + 0.375*0.25 + 0.25*0.25 + 0.25*0.25 =
       1/32 + 3/32 + 2/32 + 2/32 = 0.25

If all votes are 100% correctly decided by y1, correctness graph has 1 for all edge weights. Thus cumulative goodness
       0.25 * 1 + 0.25 * 1 + 0.25 * 1 + 0.25 * 1 = 1 = 100%
After markov iteration attains stationary distribution, cumulative goodness of all vertices also becomes stationary.

Following are the votes received by a vertex x1 from voters y1,y2,y3,y4 having respective cumulative goodness:
       y1 - x1: 0.75
       y2 - x1: 0.3 (voter with cumulative goodness < 0.5)
       y3 - x1: 0.8
       y4 - x1: 0.9

What is the probability in average case that atleast one voting vertex has cumulative goodness < 0.5? Link graph voti
```

```
Pr[atleast one adjacent vertex y to a candidate vertex x has cumulative goodness < 0.5 after markov iteration] = 1 -

From Markov inequality tail bound, Pr[cumulative goodness > 0.5] <= mean/0.5
Pr[cumulative goodness < 0.5] = 1 - Pr[goodness > 0.5] >= 1 - mean/0.5
where mean is the average cumulative goodness of all vertices in link graph.

When mean is 0.5 (uniform distribution), Pr[cumulative goodness < 0.5] >= 1 - 0.5/0.5 = 0
For any other values of mean < 0.5, Pr[atleast one adjacent vertex y to a candidate vertex x has cumulative goodness

In discrete random variable case, if goodness takes n discrete values between 0 and 1 then, mean = summation(x*p(x))
       n(n+1)/2*n*n = 0.5 + 0.5/n which tends to 0.5 when n tends to infinity.
Thus mean cumulative goodness is 0.5 in uniform distribution.

Objective and Subjective rankings can be represented on a complex plane with following notation:
       (intrinsic merit=m) + i(perception=p) = m + ip
where p is a function of m. Any text,visual or voice can be plotted on complex plane in previous notation. Ideally m

Riemann Hypothesis if true implies Re=0.5 for non trivial zeros of Riemann Zeta Function. If a set of complex number

Any k-coloring of a sequence of text and audio-visuals(AVs) denoted by integers is a classifier. 2-coloring or comple
       11110000111100001110000111000111111100000

       ----    ----    ---    ---   ------             function f:
          .....    ----   ----   ---      -----        complement g:

Vapnik-Chervonenkis Shattering or VC Shattering is defined as:
Let H be a set of sets and C be a set. C is shattered by H if H intersection C = powerset of C = 2^C. Intuitively, H

2-coloring/complementation is a classifier on real line or integer sequences. 2 colors/binary encodings/complementati

References:
-----------
420.1 Probability and Statistics with Reliability, Queuing and Computer Science Applications - Pages 223-225 - [Kishc

------------------------------------------------------------------------------------------------------------------------
421. (THEORY) Human Resource Analytics, Interview Algorithm and Intrinsic Merit - related to 314, 360 and 366 - 26 Ju
------------------------------------------------------------------------------------------------------------------------
Stability of Interview TQBF circuit which is the theoretical formalism of Interview Algorithm Intrinsic Merit has bee
(*) Construct a complete merit metric feature vector space for an individual based on past records( work and academic
(*) Construct a contraction map(s) which contracts this merit vector space into a sample subspace. This contraction m

Real life interviews typically have following usecase (e.g IT industry) - a slightly modified version of previous TQB
(*) suitability for a requirement
(*) technical discussions (question-answering on programming/projects etc.,)
(*) experience

Intrinsic Merit of Collection of Humans are measured in economics literature by many indices like GDP,Human Developme
       IPR(candidate)=geometric_mean(IPR(interview)*IPR(education)*IPR(experience))

From 359 and 365, NoiseSensitivity(InterviewTQBF) which is the dual of Stability is defined as:
       NoiseSensitivity(InterviewTQBF) = 0.5-0.5*Stability(InterviewTQBF) = 0.5-0.5*a^2*n^2/4

Error in interview is equal to NoiseSensitivity of interview TQBF. What is the probability of interview process faili
       Pr[Error in interview process > e ] <= mean/e where mean is the average error or NoiseSensitivity of interviev
       Pr[Error in interview process > e ] <= NoiseSensitivity(InterviewTQBF)/e

While suitability can be easiliy quantified for error and interview TQBF having previous error bound, experience is a
       Experience = f(number of job/academic/private hops, intrinsic merit at each previous hop, experience per hop)

Interview algorithm being a TQBF satisfiability problem is PSPACE-complete (=IP=AP). There are existing question-answ

TQBF formulation of question-answering is far stringent than traditional interviews. Existential and Universal quanti

Intrinsic merit metric space of feature vector points can be construed as a Hypergraph with feature vector points as

Similar notion of transversal hypergraph can be applied to Recursive Gloss Overlap Definition Graph too (considering

Like usual text documents, candidate resumes can be represented as either a Recursive Gloss Overlap graph or a Though
*) Graph: Resume text is mapped to a graph by Recursive Gloss Overlap algorithm. Core number based classifier brings
*) Hypergraph: Resumes are stored in ThoughtNet as Hypergraph index. Querying results in similar resumes. ThoughtNet

References:
-----------
421.1 P,NP and examinations - https://terrytao.wordpress.com/2009/08/01/pnp-relativisation-and-multiple-choice-exams/
421.2 Shrink Map and Contraction Map - [Topology - James Munkres] - pages 181-182
421.3 Banach Fixed Point Theorem - https://en.wikipedia.org/wiki/Banach_fixed-point_theorem
421.4 Davis-Putnam-Logemann-Loveland (DPLL) decision tree solver algorithm for QBF - http://personalpages.manchester.
421.5 Parallel algorithm for Hypergraph Transversals - https://people.mpi-inf.mpg.de/~elbassio/pub/COCOON05.pdf
421.6 Efficient algorithm for Hypergraph Transversals - http://jgaa.info/accepted/2005/KavvadiasStavropoulos2005.9.2.
421.7 Compendium of Intrinsic Performance Ratings in Chess - https://www.cse.buffalo.edu/~regan/papers/pdf/Reg12IPRs.
421.8 Human Development Index (New) - https://poseidon01.ssrn.com/delivery.php?ID=615100008007024087109082112068031006
```

```
7753
7754  ----------------------------------------------------------------------------------------------------------
7755  422. (FEATURE-DONE) Text Summarization from Recursive Gloss Overlap Graph Core - Commits - 28 June 2017
7756  ----------------------------------------------------------------------------------------------------------
7757  (*) Added a new function to create a summary from text - This function creates the core of a Recursive Gloss Overlap
7758  with certain core number and writes out a text sentence for each edge in core subgraph obtaining least common ancestc
7759  (*) logs for this have been committed to python-src/testlogs
7760
7761  Summarization from k-core(s) of a Recursive Gloss Overlap graph captures the most crucial areas of the text because d
7762
7763  ----------------------------------------------------------------------------------------------------------
7764  423. (FEATURE-DONE) Text Summarization from Recursive Gloss Overlap Graph Core - Commits - 30 June 2017
7765  ----------------------------------------------------------------------------------------------------------
7766  (*) New clause added to classify the text and match the prominent core number word vertices in the text and only add
7767  (*) This is an alternative to k-core subgraph traversal and creating text programmatically. It is based on the heuris
7768  capture the essence/classes the text belongs to.
7769  (*) Prominent core number classes are shaved off from the sorted core number list returned by RGO classifier and top
7770
7771  ----------------------------------------------------------------------------------------------------------
7772  424. (THEORY) Dense Subgraph Problem and Mining patterns in Graph Representation of Text - 2 July 2017
7773  ----------------------------------------------------------------------------------------------------------
7774  Recursive Gloss Overlap and Recursive Lambda Function Growth algorithms create graphs from text documents. Unsupervis
7775  prominent vertices of the definition graph and Text summarization at present depend on finding k-core subgraphs. In g
7776  problem to find Dense Subgraphs of a Graph where density of a subgraph S of a graph G is defined as:
7777         d(S) = |Edges of S| / |Vertices of S|
7778  There are polynomial time maxflow based algorithms and approximations to find dense subgraphs of a graph.
7779
7780  References:
7781  ----------
7782  424.1 Goldberg Algorithm, Charikar Algorithm and k-Cliques Densest Subgraph Algorithm for Dense Subgraph Discovery -
7783
7784  ----------------------------------------------------------------------------------------------------------
7785  425. (THEORY) Pseudorandom non-majority choice and Bounded Electorate Majority Choice - 3 July 2017 - related to 53.7
7786  ----------------------------------------------------------------------------------------------------------
7787  Bounded Electorate Majority Choice:
7788  -----------------------------------
7789  Odd Electorate with 3 voters is the minimum possible majority voting setting which vouchsafes a clear winner. This is
7790
7791  Pseudorandom Non-majority Choice:
7792  ---------------------------------
7793  Majority social choice has only one level of error probability i.e goodness for each voter while Non-majority pseudor
7794
7795  Let number of voter decision functions with goodness xi = m(xi). Total number of voters N = m(x1)+m(x2)+m(x3)+...+m(x
7796  Effective goodness of a PRG choice is the mean:
7797         1/N * summation(xi*m(xi)) = (x1*m(x1) + x2*m(x2) + x3*m(x3) + ... + xn*m(xn)) / N
7798  When all voter functions have goodness 1 then PRG choice has effective goodness 1.
7799
7800  Probability of choosing a voter of Goodness xi = m(xi)/N
7801  Conditional goodness probability of a PRG chosen voter SAT =
7802  Pr[choosing voter SAT of goodness xi] * Pr[goodness of SAT] = [m(xi)/N]*xi
7803
7804  When m(xi)*xi/N = 1, Goodness of PRG choice is 1. This can happen only if m(xi)*xi = N => m(xi) = N and xi = 1 i.e al
7805  ---------------------------------
7806  When goodness is 1 for both LHS PRG choice and RHS Bounded Electorate Majority Choice, PRG choice is a BPP/BPNC/RNC/R
7807
7808  NP in P/poly also implies PH is in P/poly. It is not known if this implies PH-complete problems exist (because PH col
7809
7810  PH-completeness proof outline:
7811  ------------------------------
7812  If NP is in BPP, NP is in P/poly because BPP is in P/poly.
7813  => If NP is in P/poly, PH is in Sigma(p,2) from Karp-Lipton-Sipser Collapse Theorem
7814  => If NP is in P/poly, PH collapses to P/poly
7815  => There are complete problems in each level k of the Polynomial Hierarchy(correspond to a k-depth QBFSAT).
7816  => There is a complete problem in Sigma(p,2) corresponding to 2-QBFSAT.
7817  => All problems in Sigma(p,2) can be reduced to this Sigma(p,2)-complete 2-QBFSAT problem.
7818  => All problems in PH collapse to Sigma(p,2) if NP is in P/poly
7819  => All problems in PH can be reduced to this Sigma(p,2)-complete 2-QBFSAT problem.
7820  => Sigma(p,2)-complete problem is thus a PH-Complete problem
7821
7822  References:
7823  -----------
7824  425.1 Proof of Karp-Lipton-Sipser collapse theorem - PH in P/poly -  http://www.cse.iitm.ac.in/~jayalal/teaching/CS68
7825
7826  ----------------------------------------------------------------------------------------------------------
7827  426. (FEATURE-DONE) Updates to Text Summarization from Dense Subgraph of Recursive Gloss Overlap graph of a Text -
7828  Commits - 4 July 2017
7829  ----------------------------------------------------------------------------------------------------------
7830  (*) import matplotlib commented in RGO classifier
7831  (*) New clause added to Text Summarization: This finds the common path between 2 word vertices synsets by inheriting
7832  existing WordNet code for shortest wordnet path distance and constructs sentences by a sliding window for each succes
7833  pair of intermediate vertices in this common path. This creates a deeper profund summary than least_common_hypernyms(
```

(*) Logs for this have been committed to testlogs/.

It has to be noted that, dense subgraph traversal and writing unguided summary sans training data mimicks human recur
comprehension and looks non-conventional for human reading. Presently only hypernym (IS A) relation is used for conne
More comprehensive humane-looking summary can be created by Meronyms (HAS A) and Holonyms(IS PART OF) in WordNet API.
presently there is no python API for ConceptNet5 and data has to be queried as RESTful JSON objects.

---

427. (FEATURE-DONE) Updates to Text Summarization - Commits - 5 July 2017

(*) Changed the class-sentence matching clause by increasing the percentile of prominent classes
(*) Changed relevance_to_text() by comparing each sentence chosen by prominent class match to the sentences in text -
Longest Common Subsequence matching difflib library function is invoked for this similarity. Sentences are chosen als
(*) Percentage of summary to the size of the text has been printed as a ratio. Logs have been committed to testlogs/

---

428. (FEATURE-DONE) Updates to Text Summarization - choosing sentences matching class labels - 6 July 2017

(*) Some experimentation on choosing relevant sentences to be added to summary was performed
(*) relevance_to_text() invocation has been changed as per algorithm below:
for each prominent dense subgraph k-core class label
find the synset definition of class label
for each sentence
invoke relevance_to_text() similarity function between sentence and the class label definition
and add to summary if the relevance ratio > 0.41 and if not already in summary
(*) Ratio 0.41 was arrived at heuristically:
- For relevance ratio 0.1, summary ratio was 0.65
- For relevance ratio 0.2, summary ratio was 0.48
- For relevance ratio 0.3, summary ratio was 0.35
- For relevance ratio 0.4, summary ratio was 0.21
- For relevance ratio 0.5, summary ratio was 0.02
There is a drastic dip in size of summary if relevance threshold is increased beyond 0.4. Because of this 0.41 has be
hardcoded.
(*) Logs for this have been added to testlogs/
(*) Summary generated is human readable - subset chosen from actual text.

---

429. (FEATURE-DONE) ConceptNet 5.4 Python RESTful API implementation - lookup, search and association - 7 July 2017

(*) This commit implements the RESTful Python requests HTTP API for querying ConceptNet 5.4 dataset
(*) Three functions for looking up a concept, searching a concept and finding similar associated concepts have been
implemented with 3 endpoints (as per the documentation in https:*//github.com/commonsense/conceptnet5/wiki/API/2349f2b*
(*) ConceptNet is quite different from WordNet in representation as JSON dictionary as against graph in WordNet.
(*) ConceptNet 5.4 endpoints have been invoked instead of 5.
(*) If necessary ConceptNet can replace all WordNet invocations in a later point in time. But such a replacement is n
probably require almost all Recursive Gloss Overlap related code to be rewritten from scratch. But that depends on ho
weighs against WordNet in measuring semantic meaningfulness.

---

430. (THEORY) Contradictions in bounds between finite and infinite electorate - 8,10,11,12,13 July 2017

Lowerbounds by equating the goodness of Non-majority and Majority social choice thus far mentioned in drafts in this
        (*) There are two possible paths to social choice - Non-majority and Majority
        (*) Each social choice belongs to a computational complexity class
        (*) Goodness of a social choice is the measure of error-free-ness of the choice made in either paths i.e deci
        (*) RHS Majority voting has been assumed to abide by Homegeneous version of Condorcet Jury Theorem convergenc
        (*) Goodness of either choice majority or non-majority must be equal
        (*) Either LHS or RHS has to be a complete problem for a complexity class C.
        (*) Traditional literature on boolean majority functions and circuits assumes that input to majority is readi
        (*) Previous Oracle access based proofs have been intentionally circumvented by replacing Oracles with Boolea
        (*) Assuming all above, LHS is an algorithm for RHS complete problem (or viceversa) creating a lowerbound.
        (*) All bounds derived in this draft assuming above do not follow conventional lowerbound techniques e.g Circ
        (*) Most importantly drafts in this document are just analyses of various social choice functions, their comp

Equality of Goodness of PRG choice and Majority voting for finite electorate of size 3:

Let x1,x2,x3 be the goodness of 3 voter SATs. PRG choice randomly chooses one of the 3 voters while majority voting i

Goodness of PRG choice:

= (x1*m(x1) + x2*m(x2) + x3*m(x3)) / 3
When all 3 have equal goodness 1, effective goodness is:
= (1*1 + 1*1 + 1*1)/3 = 1

Goodness of Majority choice:

This is the bounded version of CJT binomial series summation. When x1=x2=x3=1:
= (3C2(1)^2(0)^1 + 3C3(1)^3(0)^0) = (0 + 1) = 1

Possible Lowerbounds described previously for Unbounded and Bounded electorate lead to contradictions as below:
(*) In infinite voter case, homogeneous voter CJT circuit in BPP is derandomized to P if CJT converges => P=BPP.

```
(*) In finite voter case, NP-complete RHS has BPP algorithm in LHS (NP in BPP) implying NP in P/poly and collapse of

Contradiction 1 :
-----------------
If BPP=P (unbounded CJT) and NP is in BPP (bounded) then NP is in P=BPP => P=NP. This conflicts with P != NP implied
        (*) the Majority Hardness Lemma (318),
        (*) Polytime learning of NP implying NP does not have polytime algorithms (368) and
        (*) HLMN PARITYSAT counterexample (53.15)
but concurs with :
        (*) high percentage of random k-SATs satisified in Approximate CNF SAT solver by least squares (376).

But can infinite voter CJT circuit be in BPP and thus in P/poly? Infinite voter CJT circuit is of polynomial size if

Contradiction 2 :
-----------------
If NP is in BPP and thus in P/poly (irrespective of BPP=P), similar conflicts arise. An assumption made in compositic

Reference 430.3 suggests there exists a random oracle A relative to which NC^A is in P^A. P having NP oracle access i

Replacing oracles with compositions and applying depth lowerbounds for Majority+VoterSAT circuit composition as below
       D(Maj + Voter) = CommunicationComplexity(R(Maj,Voter)) >= 5.3*logm  + n - O(m*logm/n)
does away with the hassles of relativization and directly gives the depth lowerbound of the majority voting circuit c

Example delta(p,2)-complete problem mentioned in 430.5:
------------------------------------------------------
While number of queries to NP oracle <= k
{
       (*) Query a 3SAT oracle for a satisfying assignment for a 3CNF Voter SAT
       (*) Store the queried satisfying assignment in a linked list in sorted order - this is linear per insertion i.
}
Output 1 if last element in the list ends with binary 1 digit.

This algorithm makes k queries to an NP oracle and has O(k^2) time complexity and thus in delta(p,2). Hardness follow

Mapping above delta(p,2) complete problem to proving completeness of NC1^NP majority voting is non-trivial. Majority

But this also implies that any Majority^A for a random oracle A is not complete if Majority is not complete for NC1.

Case 1 - s = 2^n:
----------------
This makes the new majority circuit sans oracle to be of depth O(n*log2+c*log(n)) which is not polylog depth and poly

Case 2 - s = n^c:
-----------------
This new majority circuit has depth O(2*c*log(n)) and size O(n^2c) and is obviously in NC1 and computes majority. But


References:
-----------
430.1 Definition of Homogenenous Voter - http://www.uni-saarland.de/fak1/fr12/csle/publications/2006-03_condorcet.pdf
430.2 Counting Classes and Fine Structure between NC and L - [Samir Datta , Meena Mahajan , B V Raghavendra Rao , Mic
430.3 For a random oracle A, NC^A is strictly contained in P^A - https://complexityzoo.uwaterloo.ca/Complexity_Zoo:N,
430.4 Delta(p,2) - P has NP oracle - https://complexityzoo.uwaterloo.ca/Complexity_Zoo:D#delta2p
430.5 Delta(p,2) complete problem - https://complexityzoo.uwaterloo.ca/Zooref#kre88 - [Krentel] - Given a Boolean for
430.6 Spira theorem - any formula of leaf size s can be transformed into a formula of depth log(s) - https://www.math
430.7 Size of a circuit composition - alternative form of KRW conjecture - http://www.math.ias.edu/~avi/PUBLICATIONS/
430.7 Size of a circuit composition - alternative form of KRW conjecture - http://www.math.ias.edu/~avi/PUBLICATIONS/
430.8 Log Depth circuits for Division and Related - [BeameCookHoover] - https://pdfs.semanticscholar.org/29c6/f0ade6c
430.9 NC reducibility - reducing one NC instance to another - https://homes.cs.washington.edu/~ruzzo/papers/limits.pd
430.10 Catechism on open problems - Interview of Eric Allender - https://books.google.co.in/books?id=7z3VCgAAQBAJ&pg=
430.11 Parallel Computation and the NC hierarchy relativized - [Christopher Wilson] - https://link.springer.com/chapt

-------------------------------------------------------------------------------------------------------------------
431. (THEORY) Theoretical Formalism for an Electronic Voting Machine based on Locality Sensitive Hashing - 13 and 14
-------------------------------------------------------------------------------------------------------------------
Locality Sensitive Hashing and their relevance to Multiway Contests have been described in 319. Voters voting for sam

Non-boolean social choice functions which a voter computes to obtain a candidate index have been described previously
       (*)  receives voters in a streaming sequence,
       (*)  each voter solves a non-boolean SAT (which could be in an arbitrary complexity class) having oracle acces
       (*)  oracle returns a candidate index and counter is incremented
       (*)  LSH based voting has a special step - it compares two voters for similarity i.e if their oracle queries r
       (*)  Multipartisan SAT Oracle internally has to implement the following:
               - Iterate through all candidates
               - Find the maximum number of clauses that can be satisfied by each candidate and quantify it as score
               - Sort the scores for the candidates and return the topranked candidate.
       (*) Sort the LSH by length of buckets
       (*) MAXSAT has been used in lieu of Exact SAT for grading the candidates based on number of clauses satisfied.
       (*) Random k-CNF SAT Solver implemented in 276 approximates by solving system of equations by least squares.
       (*) Boolean Majority function is a special case of multipartisan voting: Restrict the number of candidates to
       (*) Voter SAT Oracles could be Constraint Satisfaction Problem(CSP) Solvers too which allow reals. This has be
```

Circuit Value Problem finds if a circuit encoding evaluates to 1 or 0 for an input assignment. This is equivalent to
(*) Voter has a Constraint Satisfaction Problem (boolean or real)
(*) Candidate has an assignment to the variables of voter CSP (Prover)
(*) Voter verifies the assignment to CSP (Verifier)
This kind of Circuit Value Problem (CVP) formulation has been avoided throughout drafts in document for Majority Voti

References:

431.1 Locality Sensitive Hashing - [Alex Andoni] - http://web.mit.edu/andoni/www/LSH/index.html
431.2 Lowerbounds for Locality Sensitive Hashing - [MotwaniAssafPanigrahi] - http://theory.stanford.edu/~rinap/papers
431.3 Optimal lowerbounds for Locality Sensitive Hashing - [ODonnell] - https://www.cs.cmu.edu/~odonnell/papers/lsh.p
431.4 Reflections on Trusting Trust - Trojan horses in code - program that prints itself - Quines - Godel,Escher,Bach
431.5 Problems with Electronic Voting Machines (e.g DRE machines), VVPAT and Trusting source code - [Bruce Schneier]
431.6 B-Cryptographic counters protocol augmented with Public Key Infrastructure for Incrementing Counters in Tabulat
431.7 PP is as hard as polynomial time hierarchy - [Toda] - http://epubs.siam.org/doi/abs/10.1137/0220053?journalCode

---
432. (FEATURE-DONE and THEORY) Approximate CNF SAT Solver Update - percentage of clauses satisfied - 18 July 2017

(*) This commit prints percentage of clauses satisfied for each random CNF 3SAT formula for both 0 and 1 evaluations
(*) logs for ~1000 iterations of random 3SAT have been committed to testlogs/
(*) Interestingly, even failing formula assignments by least squares satisfy more than 75% of clauses of the formula
(*) Percentage of formulas satisfied by least squares heuristic is ~70%
(*) This is probably a demonstration of complement form of Lovasz Local Lemma (LLL) for MAXSAT which is:
if events occur independently with a certain probability, there is a small non-zero probability that none of them wil
(*) Dual of LLL for this MAXSAT approximation is:
if each clauses are satisfied with a certain probability, there is a large probability that all of them are satisfied

References:

432.1 Lovasz Local Lemma - https://en.wikipedia.org/wiki/Lov??sz_local_lemma

---
433. (THEORY) MAXSAT ranking of text documents, Approximate CNF SAT solver and Lovasz Local Lemma - related to 432 -
19 July 2017

Considering a set of text documents, ranking function r(X,A), is the subjective perception measure of a text X
with access to a perception oracle A:
r(X,A) = subjective rank of X as perceived by an oracle A
In the context of web link graphs of **HTML** documents, every adjacent vertex of a node N is the perceiver of N and for
adjacent vertices A and B of N:
r(X,A) = r(X,B) or r(X,A) != r(X,B)
Perception ranking of a node need not be decided just by adjacent vertices at the end of random walk markov iteration

Ranking of texts can be thought of as MAXSAT problem: Each reader of a text has a CNF (subjective) or all readers hav

Lovasz Local Lemma:
Let a1,a2,a3,...,an be set of events. If each event occurs with probability < p, and dependency digraph of these even
Pr[/\ !ai] > 0 , if ep(d+1) < 1 for e=2.7128...

CNF formula can be translated into a graph where each vertex corresponds to a clause and there is an edge between any

From LLL, If p < 1/[e(d+1)] is the probability of an event (not satisfying a clause) :
Probability that clause is satisfied: 1-p > (1-1/[e(d+1)])
Probability that all of the clauses are satisfied > (1-1/[e(d+1)])^(d+1) (for d+1 clauses)
Probability that none of the clauses are satisfied < 1-(1-1/[e(d+1)])^(d+1)
In previous example d+1 = 14 and thus probability that none of the clauses are satisified for any random CNF is:
= 1-(1-1/[e(14)])^(14) < ~31.11%
and probability of all the clauses being satisified for any random CNF is:
(1-1/[e(14)])^(14) > ~68.89%

Experimental iterations of SAT solver converge as below (10000 CNFs):
Percentage of CNFs satisfied so far: 71.14
Average Percentage of Clauses per CNF satisfied: 97.505

This coincides with LLL lowerbounds above and far exceeds it because outdegree (common literals) is less and average

---
434. (FEATURE-DONE) Approximate CNF SAT Solver update - Commits 1 - 19 July 2017

(*) Average percentage of clauses satisfied per CNF is printed at the end of 10000 iterations
(*) AsFer Design Document updated for Lovasz Local Lemma analysis of least squares CNF SAT solver
(*) logs for least square assignment of 10000 random CNFs committed to testlogs/
---
435. (FEATURE-DONE) Recursive Gloss Overlap Graph Classifier update - Commits 2 - 19 July 2017

(*) printed Betweenness Centrality (BC) of the definition graph of a text
(*) BC of a node is the ratio of number of (s-t) shortest paths going through the node to all pairs shortest paths ar
thus measures how central a node is to the graph and thus is another basis for classifying a text apart from dense su

```
8077 ------------------------------------------------------------------------------------------------------------------------
8078 436. (FEATURE-DONE) Recursive Gloss Overlap graph classifier update - commits 1 - 20 July 2017
8079 ------------------------------------------------------------------------------------------------------------------------
8080 (*) Following centrality measures for the definition graph have been printed in sorted order and top vertices are com
8081        1) Betweenness Centrality
8082        2) Closeness Centrality
8083        3) Degree Centrality
8084        4) Eigenvector Centrality (PageRank)
8085        5) Core numbers Centrality (k-core dense subgraphs)
8086 (*) While first 3 centrality measures are reasonably equal in terms of ranking central vertices, PageRank centrality
8087  slightly different
8088 (*) But all 5 centrality measures reasonably capture the central purport of the text i.e the text on
8089  "Chennai Metropolitan Area Expansion to 8848 sqkm" is classified into "Area" which is the topranked central vertex
8090
8091 ------------------------------------------------------------------------------------------------------------------------
8092 437. (FEATURE-DONE) Approximate SAT Solver update - commits 2 - 20 July 2017
8093 ------------------------------------------------------------------------------------------------------------------------
8094 (*) Increased number of clauses and variables to 16 and iterated for ~3100 random CNF formulae.
8095 (*) Following are the MaxSAT percentages after ~3100 iterations:
8096 Percentage of CNFs satisfied so far: 69.1020276794
8097 Average Percentage of Clauses per CNF satisfied: 97.6122465401
8098 (*) Previous values are quite close to 14 clauses and variables iterations done previously.
8099
8100 Lovasz Local Lemma bound for 16 clauses all having common literals:
8101 = (1-1/[e(16)])^(16)  > ~68.9233868%
8102 and average percentage of clauses satisfied after 3100 iterations far exceeds this bound at 97.61%
8103
8104 Commercially available SAT solvers (exact NP-complete decision tree evaluation based) usually scale to millions of cl
8105 this approximate polynomial time SAT solver does not have similar advantage, yet the huge percentage of clauses satis
8106
8107 Random number generator used in generating random CNF clauses is based on /dev/random and hardware generated entropy.
8108
8109 References:
8110 -----------
8111 437.1 Linux Pseudorandom Generator - https://eprint.iacr.org/2012/251.pdf
8112
8113 ------------------------------------------------------------------------------------------------------------------------
8114 438. (FEATURE-DONE) Approximate SAT Solver update - 24 July 2017
8115 ------------------------------------------------------------------------------------------------------------------------
8116 (*) Increased number of variables and clauses to (20,20).
8117 (*) Changed SciPy lstsq() to NumPy lstsq() because NumPy has a faster lstsq() LAPACK implementation as against xGELSS
8118 (*) Logs for 100 random CNFs have been committed to testlogs/ and Percentage CNFs and Clauses satisfied are as below:
8119 Percentage of CNFs satisfied so far: 58.4158415842
8120 Average Percentage of Clauses per CNF satisfied: 97.1782178218
8121
8122 ------------------------------------------------------------------------------------------------------------------------
8123 439. (THEORY) Tight Hamiltonian Cycles and Independent Sets in ThoughtNet Hypergraph for Social Networks - related to
8124 421 and other sections on ThoughtNet - 25 July 2017 and 28 July 2017
8125 ------------------------------------------------------------------------------------------------------------------------
8126 Theoretical description of ThoughtNet Hypergraph as basis for evocative thought and text analysis is mentioned previo
8127
8128 Ranking people in social network can not be done in the same way as documents are ranked because of intrinsic merit a
8129
8130 Reference:
8131 -----------
8132 439.1 Tight Hamiltonian Cycles in 3-uniform Hypergraphs - [Rodl-Rucinski-Szemeredi] theorem - https://web.cs.wpi.edu/
8133 439.2 Bianconi-Barabasi Social Network Model - https://en.wikipedia.org/wiki/Bianconi%E2%80%93Barab%C3%A1si_model - "
8134 439.3 Experience versus Talent shapes the web - http://www.pnas.org/content/105/37/13724.full - following simplistic
8135        Let M be the natural ability/merit/fitness of a vertex and E be the experience.
8136
8137        Rate of change of experience of a social profile vertex is proportional to its intrinsic fitness and present
8138        Previous assumption is a heuristic on real-life experiences - intrinsic fitness is a constant while experienc
8139        Thus at any time point, experiential learning depends on how naturally meritorious a person is and how much p
8140        helpful in increasing experiential learning. (Might have some theoretical basis in Mistake bound in computati
8141        Following formalises how a boolean function is learnt incrementally from a dataset over time by making mistak
8142
8143        dE/dt = kME
8144        => dE = kME.dt
8145        => dE/E = kM.dt
8146        => log E = kMt + c
8147        => E = e^(kMt+c)
8148        => E = e^c * e^(kMt)
8149        => Experience is exponentially related to talent/merit/fitness
8150        => People with high natural ability amass same experience in less time compared to people with low merit
8151 At time t=0, E = e^c which is equal to natural ability/merit/fitness M of a social profile vertex in Bianconi-Barabas
8152 Therefore, E = M*e^(kMt). At t=0, experience equals merit and grows exponentially over time. But how to measure the f
8153 439.4 Bose-Einstein Condensation in Complex Networks - [Bianconi-Barabasi] - Physical Review Letters - http://barabas
8154 439.5 PAFit - Joint estimation of preferential attachment and node fitness in growing complex networks - [Thong Pham,
8155
8156 ------------------------------------------------------------------------------------------------------------------------
8157 440. (FEATURE-DONE) Scheduler Analytics for VIRGO Linux Kernel - Commits - 31 July 2017 and 1,3 August 2017
```

```
8158 ----------------------------------------------------------------------------------------------------------------------
8159 (*) As mentioned in NeuronRain FAQ at http://neuronrain-documentation.readthedocs.io/en/latest/, analytics driven lin
8160 is the ideal application of machine learning within kernel. There are two options to go about:
8161 (*) First is Application Layer Scheduling Analytics in which a python code for example, learns from linux kernel logs
8162 and /proc/<pid>/sched data of CFS and exports key-value pairs on how to prioritize processes' nice values in /etc/ker
8163 (*) Second is Kernel Layer Scheduling Analytics in which kernel has to make upcalls to userspace machine learning fun
8164 (*) Third option is to implement a separate kernel module with access to process waiting queue which loops in a kerne
8165
8166 First option is chosen for time being because:
8167         - it is top-down
8168         - it does not interfere in existing scheduler code flow much
8169         - minimal code change is required in kernel scheduler e.g as a KConfig build parameter and can be #ifdef-ed
8170         - key-value pair variable mechanism is quite flexible and broadcast kernelwide
8171         - any interested module in kernel can make use of them not just scheduler
8172         - it is asynchronous while kernel layer scheduling is synchronous (scheduler has to wait to know priority for
8173         - second and third options are circuitous solutions compared to first one.
8174         - Languages other than C are not preferable in kernel (http://harmful.cat-v.org/software/c++/linus)
8175         - Key-Value pair protocol between userspace and kernelspace effectively tames language barriers between user
8176         - Key-Value storage can also exist on a cloud too not necessarily in /etc/kernel_analytics.conf file thus tra
8177         - first option is more suited for realtime kernels
8178
8179 ##############################################################################################
8180 #Scheduler Analytics for Linux Kernel:
8181 #====================================
8182 # Following DeepLearning models learn from process perf variables - CPU, Memory, Context switches, Number of Threads
8183 # for each process id retrieved by psutil process iterator - BackPropagation, RecurrentLSTM
8184 # RecurrentGRU and ConvolutionNetwork models learn software analytics neural networks from psutil process performance
8185 # Key value pairs learnt from these can be read by Linux Kernel Scheduler or anything else
8186 # and suitably acted upon for changing the process priorities dynamically. Any kernel module can make an upcall to th
8187 # executable and dump the key-value pairs in /etc/kernel_analytics.conf. Presently the implementation is quite primit
8188 # classifies the output layer of neural network into "Highest, Higher, High, Normal, Medium, Low, Lower and Lowest" p
8189 # in the format: <pid#deeplearningalgorithm>=<scheduled_priority_class>. Number of iterations has been set to 10 for
8190 ##############################################################################################
8191
8192 Following is a simulation of how the key-value pair for repriotization from Scheduler Analytics could affect the kern
8193        (*) Linux kernel scheduler has the notion of jiffies - HZ - set of clock ticks - for time slicing processes
8194        (*) Each priority class has a red-black binary search tree based queue - deletion is O(1) and insertion is O(l
8195        (*) On receipt of repriotization analytics, kernel_analytics exports it kernelwide, scheduler reads this <pid>
8196        (*) Thus multiple red-black trees make a trade-off and rebalance periodically with no manual nice intervention
8197        (*) Every repriotization is therefore O(logN).
8198        (*) In realtime bigdata processing of heavy load, priorities can change quite frequently. For m reprioritizati
8199
8200 References:
8201 -----------
8202 440.1 Scheduler Activations - Effective Kernel Support for the user level management of parallelism - http://dl.acm.o
8203 440.2 Controlling Kernel Thread Scheduling From Userspace - http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.
8204
8205 ----------------------------------------------------------------------------------------------------------------------
8206 441. (THEORY) Social Networks, Money Flow Markets, Pricing Equilibrium, Centrality, Intrinsic versus Perceptive Valua
8207 and 11 August 2017
8208 ----------------------------------------------------------------------------------------------------------------------
8209 Flow of money in a network of buyers and sellers and pricing have been described in KingCobra design document - https
8210 In a flow market digraph,there is a directed edge from a vertex b to vertex s of weight p where b is a buyer, s is a
8211
8212 In money flow markets, each buyer i is allocated a good/service j of x(ij) units with utility u(ij) with price p(j).
8213
8214 Previous example is mooted to invoke the striking parallels between buyer versus seller price equilibrium and intrins
8215 Convex Program for Market Equilibrium is defined as below:
8216        Maximize u(ij)^e(i) where buyer i has money e(i) at disposal
8217 subject to:
8218        Total happiness of a customer = Sigma(u(ij)*x(ij)) where u(ij) is the utility buyer i gets from x(ij) units of
8219
8220 Following maps a Market to Merit equilibrium:
8221 Intrinsic merit of a (corresponds to Seller) vertex is defined by a feature vector of merit variables. Buyers are per
8222        Maximize u(ij)^e(i) where perceiver i has total perceived merit e(i) at disposal and u(ij) is the utility perc
8223 subject to:
8224        Total happiness of a perceiver = utility of i = Sigma(u(ij)*x(ij)) where u(ij) is the utility perceiver i gets
8225        and p(j) is the equilibrium perceived merit of a variable j
8226 Solving the previous convex program results in an optimal merit vector (set of p(j)'s) and weights of merit variables
8227
8228 This equilibrium is closely related to Nash Bargaining Problem which postulates existence of an agreement and disagre
8229
8230 References:
8231 -----------
8232 441.1 Algorithmic Game Theory - Chapter 5 and Chapter 15 - Existence of Flow Markets Equilibrium and Nash Bargaining
8233 441.2 New Convex Program for Fisher Market Equilibria - Convex Program Duality, Fisher Markets, and Nash Social Welfa
8234 441.3 Triangle Inequality based Christofides 1/2-approximation Algorithm for Travelling Salesman Problem - Combinator
8235
8236 ----------------------------------------------------------------------------------------------------------------------
8237 442. (FEATURE-DONE) Scheduler Analytics update - commits - 1 August 2017
8238 ----------------------------------------------------------------------------------------------------------------------
```

```
(*) Updated AsFer Design Document
(*) Updated python-src/software_analytics/DeepLearning_SchedulerAnalytics.py to write in /etc/kernel_analytics.conf
(*) logs committed to testlogs/

------------------------------------------------------------------------------------------------------------------------
443. (FEATURE-DONE) Graph Density (Regularity Lemma) and Bose-Einstein Fitness implementations - commits - 1 August 2
------------------------------------------------------------------------------------------------------------------------
(*) New intrinsic merit measures based on graph density (regularity lemma) and Bose-Einstein intrinsic fitness have b
and invoked
(*) logs have been committed to testlogs/

------------------------------------------------------------------------------------------------------------------------
444. (THEORY) Mistake bound learning(MB), Experience and Intrinsic merit, Social Network Analytics - 2,3,11 August 20
------------------------------------------------------------------------------------------------------------------------
Mistake bound learning, iteratively refines and converges from an initial hypothesis based on mistakes. PAC learning

Let M be the intrinsic merit (correctness) of a boolean function hypothesised in the form of conjunctions and disjunc
of experience(E) versus merit(M) as function of time(t):
       dE/dt = kME
       => dE = kME.dt
       => dE/E = kM.dt
       => log E = kMt + c
       => E = e^(kMt+c)
       => E = M*e^(kMt)

If dE/dt is replaced by number of mistakes b1 made and corrected per time unit in MB model and E is replaced by total
       => E = M*e^(kMt)
       => E = M*e^(kM*b2/b1)
Previous expression for experience depends both on intrinsic merit and mistakes made which is exactly required in hum

Traditionally mistake bounds are used only in learning boolean functions. Generalizing it to any functions and humans
       => E = M*e^(kM*g(IQ,EQ)/f(IQ,EQ))
If M=h(IQ):
       => E = h(IQ)*e^(k*h(IQ)*g(IQ,EQ)/f(IQ,EQ))

It has to be mentioned here f is nothing but the first temporal derivative of g evaluated at time t i.e f = dg/dt.
       => E = h(IQ)*e^(k*h(IQ)*g(IQ,EQ)/g'(IQ,EQ))

Alternatively, f and g can be substituted directly if known and integrable:
       g'(IQ,EQ) = k*h(IQ)*g(IQ,EQ)
       g'(IQ,EQ) = k*h(IQ)*g(IQ,EQ)
       log g(IQ,EQ) = k*M*t + c
       g(IQ,EQ) = h(IQ)*e^(k*h(IQ)*t)

Curiously, function g which depends on EQ has no equivalent in RHS except time duration t and a constant k. This is p
       g(IQ1,EQ1) = h(IQ1)*e^(k*h(IQ1)*t1)
       g(IQ2,EQ2) = h(IQ2)*e^(k*h(IQ2)*t2)
Disclaimer: Previous model of merit based on IQ and EQ are purely presumptive and disputable because of lack of well-

This model is quite generic and should apply to any computable function including boolean learnt by mistake bounds. F

Mistakes in the social networks context are proportional to bounded rationality of the human social profile vertices

Thus following equivalence is conspicuous: Emotional Quotient (proportional to) Bounded Rationality (proportional to)

BackPropagation is a mistake bound deep-learning algorithm refining a perceptron by trial and error iterations. Previ
At time x, tree reaches level l(x). Each node in the tree is a corrected mistake and each mistake corrected at level

Caveat is the previous recursive mistake correction tree is not traditional bottom-up evaluation circuit but is top-d
       M=<m1,m2,m3,...,mn> where each mi is the weight for a merit variable
For example, Discrete Probability distribution for merit/least energy/fitness can be defined as:
       Pr(M=mi) = mi/|M| where |M| is the L1 norm of M
       Sigma(Pr(M=mi)) = (m1 + m2 + m3 + ... mn) / |M| = 1
Continuous Probability distribution is the polynomial passing through above merit weight points.

References:
-----------
444.1 Mistake bound learning - https://www.cs.utexas.edu/~klivans/lec1.ps - In this teacher/learner model, learner it
444.2 Learning quickly when irrelevant attributes abound - [Nick Littlestone] - http://citeseerx.ist.psu.edu/viewdoc/
444.3 Bounded Rationality and Satisficing - https://en.wikipedia.org/wiki/Bounded_rationality
444.4 Multiplex Networks with Intrinsic Fitness: Modeling the Merit-Fame Interplay via Latent Layers - [Babak Fotouhi
444.5 Relation between Intrinsic Merit of two vertices and probability of creating link between them - Page 27 - Equa
444.6 First to Market is not Everything: an Analysis of Preferential Attachment with Fitness - [Christian Borgs, Jenn
       log(dv(t)) = cf*logt
       f=log(dv(t))/clog(t)
       Since f=M, E = M*e^(kMt) = log(dv(t)) * e^(klog(dv(t))*t/clogt) / clogt
444.7 Degree distribution and Intrinsic Fitness - https://www.cs.upc.edu/~CSN/slides/08network_dynamics.pdf - "... Co
444.8 Ghadge et al Model of Intrinsic Fitness of a vertex - https://www.cs.umb.edu/~duc/publications/papers/ijpeds10.
444.9 Vertex Intrinsic Fitness in Social Networks - [Servedio,Caldarelli] - https://arxiv.org/pdf/cond-mat/0309659.pd
444.10 Bibliography listed in Social Networking: Mining, Visualization and Security - [Mrutyunjaya Panda, Satchidanda
444.11 Vertex Fitness defined in terms of Unfitness - Network growth models: A behavioural basis for attachment propo
```

```
444.12 A ??-deformed Model of Growing Complex Networks with Fitness - [Massimo Stella , Markus Brede] - https://arxiv
444.13 Various Intrinsic Fitness Models - Barabasi-Albert, Barabasi-Bianconi, Ghadge Log Normal Fitness Attachment, C

-----------------------------------------------------------------------------------------------------------------------
445. (FEATURE-DONE) Scheduler Analytics Update - Commits - 10 August 2017
-----------------------------------------------------------------------------------------------------------------------
(*) There is a problem in defining expected priorities in BackPropagation Neural Network output layer. This is necess
weight update to happen iteratively and to quantify error between expected and actual neural network output
(*) Output layer of BackPropagation is an indicator of process priority learnt by the perceptron.
(*) Previously expected priorities were hardcoded. This has been changed as below:
        (*) Set of processes currently running are clustered and stored in an input JSON file
        (*) This JSON dictionary has key-value pairs mapping a process class to its apriori expected priority
        (*) Scheduler Analytics loads this JSON and looks up the expected priority for each process executable by its
        (*) Presently process class is just a substring of the executable name
        (*) BackPropagation does string match of this process class to executable name and finds the expected priorit
        (*) BackPropagation iteration updates the weights of all 3 layers
(*) Defining apriori class of a process is subjective and instead of substring of executable, an independent clusteri
suitable distance function can be used to write the JSON classification of processes. For example, unix process group
 processes.
(*) Previous generation of JSON classes for processes and backpropagation weight update iteration is the first phase
(*) In the second phase, weight-updated BackPropagation Neural Network takes as input runtime process statistics (CPU
to predict the actual priority of a process belonging to the apriori class.
(*) This is a depth 2 tree evaluation:
        Static - apriori expected class of a process and BackPropagation iteration
        Dynamic - fine grained priority of a process within the previous static class by evaluating the multilayer ne

Presently, input variables are per process CPU percentage, Memory percentage and ratio of involuntary context switche

-----------------------------------------------------------------------------------------------------------------------
446.(FEATURE-DONE) Celestial Pattern Mining Update - specific to NeuronRain Research in SourceForge repositories - re
-----------------------------------------------------------------------------------------------------------------------
(*) Added a documentation text file asfer-docs/NeuronRain_Research_Celestial_Pattern_Mining.txt describing how astron
mining is done and code involved therein.

(*) In python-src/autogen_classifier_dataset/MaitreyaToEncHoroClassified.py upgraded Maitreya's Dreams (http://www.sa
ephemeris text client version to 7.1.1

(*) Created separate string encoded astronomical data files for Earthquakes(USGS 100 year earthquake dataset) and
Storms(NOAA 100 year HURDAT2 dataset) Datasets from python-src/autogen_classifier_dataset/MaitreyaToEncHoroClassified
(zodiacal does not have ascendant, while ascrelative is rotated string relative to ascendant. Ascendant is defined as
 degree of the zodiac sign rising at the time of an event in eastern horizon)

(*) Executed python-src/SequenceMining.py on these 2 datasets. The results have been written to python-src/MinedClass

(*) python-src/autogen_classifier_dataset/asfer_dataset_segregator.sh has updated asfer relative location

-----------------------------------------------------------------------------------------------------------------------
447. (FEATURE-DONE) Celestial Data Pattern Mining - Updates to autogen_classifier_dataset pre-processing and classifi
-----------------------------------------------------------------------------------------------------------------------
(*) An 8 year old lurking bug in cpp-src/NaiveBayesClassifier.cpp has been resolved.It was causing a null class to be
bayesian argmax() computation.
(*) Lot of new articles on earthquakes and hurricanes have been added as training dataset for NaiveBayesian Classifie
have lot of research information on predicting hurricanes and earthquakes
(*) python-src/autogen_classifier_dataset/AsferClassifierPreproc.py has been updated to include new articles in train
(*) words.txt,word-frequency.txt,training-set.txt,topics.txt,test-set.txt have been re-created by executing python-sr
(*) Datasets are classified into python-src/autogen_classifier_dataset/EventClassDataSet_Earthquakes.txt and python-s

-----------------------------------------------------------------------------------------------------------------------
448. (FEATURE-DONE) Updated Ephemeris Search Script - Commits - 24 August 2017
-----------------------------------------------------------------------------------------------------------------------
Updated mined class association rule parsing in python-src/MaitreyaEncHoro_RuleSearch.py

-----------------------------------------------------------------------------------------------------------------------
449. (FEATURE-DONE) Approximate CNFSAT Solver update - SciPy Sparse - 28 August 2017
-----------------------------------------------------------------------------------------------------------------------
(*) imported scipy.sparse.linalg least squares function lsqr().
(*) Number of clauses and variables set to 18 each.
(*) After 673 iterations of Random 3CNFs following satisfied clauses data were observed:
Percentage of CNFs satisfied so far: 58.7537091988
Average Percentage of Clauses per CNF satisfied: 97.0161556215

From LLL, If p < 1/[e(d+1)] is the probability of an event (not satisfying a clause) :
Probability that clause is satisfied: 1-p > (1-1/[e(d+1)])
Probability that all of the clauses are satisfied > (1-1/[e(d+1)])^(d+1) (for d+1 clauses)
Probability that some of the clauses are satisfied < 1-(1-1/[e(d+1)])^(d+1)

If all clauses overlap:
In previous example d+1 = 18 and thus probability that all of the clauses are satisified for any random CNF is :
(1-1/[e(18)])^(18)  > ~68.96%
and probability that some but not all of the clauses being satisified for any random CNF is:
1-(1-1/[e(18)])^(18)  < ~31.043%
```

```
8401
8402  If none of the clauses overlap:
8403  d=0 and probability that all the clauses are satisfied  (1-1/e) > ~63.21%
8404  and probability that some but not all of the clauses being satisified for any random CNF is  < ~36.79%
8405
8406  ------------------------------------------------------------------------------------------------------------------------
8407  450. (THEORY) Analysis of Error Upperbound for Approximate CNFSAT Solver based on Lovasz Local Lemma clause dependenc
8408  ,30 August 2017
8409  ------------------------------------------------------------------------------------------------------------------------
8410  Probability of a clause being not satisfied from Lovasz Local Lemma:
8411          p < 1/(e(d+1))
8412  Probability of a clause being satisfied:
8413          1-p > 1 - 1/(e(d+1))
8414  Probability of all n clauses being satisfied (ExactSAT):
8415          (1-p)^n > [1 - 1/(e(d+1))]^n
8416  Probability of some or none of the clauses being satisfied:
8417          1-(1-p)^n < 1 - [1 - 1/(e(d+1))]^n
8418  If there are no overlaps, d=0:
8419          Probability of all n clauses being satisfied = (1-p)^n > [1 - 1/e]^n = [1 - 1/e]^n
8420          Probability of some or none of the clauses being satisfied = 1 - (1-p)^n < 1 - [1 - 1/e]^n
8421
8422  Probability of atleast k of n clauses being satisfied:
8423          Pr[#SATclauses >= k] = Pr[#SATclauses = k] + Pr[#SATclauses = k+1] + Pr[#SATclauses = k+2] + ... + Pr[#SATcla
8424
8425          = (1-p)^k + (1-p)^(k+1) + ... + (1-p)^n
8426
8427  if q = 1-p:
8428          = q^k + q^(k+1) + ... + q^n
8429          = q^k * (1 + q + q^2 + ... + q^(n-k))  [Geometric series]
8430          = q^k (1 - q^(n-k+1)) / (1-q)
8431  But from LLL, p < 1/e(d+1)
8432          q = 1 - p > (ed+e-1)/(ed+e)
8433
8434  Substituting for q:
8435          [(ed+e-1)/(ed+e)]^k [1 - [(ed+e-1)/(ed+e)]^(n-k+1)]
8436          ---------------------------------------
8437                  1 - [(ed+e-1)/(ed+e)]
8438
8439  ######################################################################
8440  #Probability of atleast k of n clauses being satisfied:         #
8441  #       [ed+e-1]^k [(ed+e)^(n-k+1) - [ed+e-1]^(n-k+1)]          #
8442  #       ---------------------------------------                 #
8443  #                [(ed+e)^n]                                     #
8444  ######################################################################
8445
8446  if k=n:
8447          (ed+e-1)^n / (ed+e)^n
8448
8449  If overlap of variables across the clauses is huge, ed+e-1 ~ ed+e
8450  Maximum overlap d is (n-1) i.e a clause has common variables across all other clauses
8451
8452  When d is maximum:
8453  ------------------
8454  Lt (d -> n-1, n -> infinity) ((ed+e-1)/(ed+e))^n
8455          = [((e(n-1)+e-1)/(e(n-1) + e)]^n
8456          = [(en-e+e-1)/(en-e+e)]^n
8457          = [(en-1)/(en)]^n
8458          = [(1-1/ne)]^n
8459          = [1 - 0.3678/n]^n
8460  For infinite n:
8461          = e^(-0.3678)
8462  Pr[#SATclauses = n]     = 69.22%
8463
8464  When overlap is absent at d=0:
8465  ------------------------------
8466  Pr[#SATclauses = n]     = (e-1)^n/e^n = (0.6321)^n -> 0 for huge n
8467
8468  When d=1 (minimum overlap):
8469  ---------------------------
8470  Pr[#SATclauses = n] = [2e-1]^n / [2e]^n = [1-0.5/e]^n = (0.81606)^n -> 0 for huge n
8471
8472  => When the number of variables overlapping across clauses tends to a huge number almost equal to number of clauses,
8473  linear equations obtained by relaxing each of the clauses are solved by least squares and probability of all clauses
8474  to 69.22%.  High overlap of variables across clauses is the most probable occurrence in real world SAT solvers. If th
8475          Expected value of d = 1/(n-1) * (n-1)n/2 = n/2
8476
8477  For mean overlap of n/2 variables across clauses:
8478  ######################################################################
8479  #Probability of atleast k of n clauses being satisfied:         #
8480  #       [en/2+e-1]^k [(en/2+e)^(n-k+1) - [en/2+e-1]^(n-k+1)]    #
8481  #       ---------------------------------------                 #
```

```
#                [(en/2+e)^n]                                  #
#################################################################

When k=n:
        [en/2+e-1]^n
        ------------
        [en/2+e]^n

        = [1 - 0.7357888/(n+1)]^n > [1 - 0.7357888/n]^n
        = Lt (n->infinity) [1 - 0.735758882/n)]^n = e^-0.73578882
        => Pr[#SATclauses = n] > 47.91%

=> When the number of variables overlapping across clauses are uniformly distributed, Probability of all clauses beir
3CNF is lowerbounded as 47.91% when number of clauses tend to infinity.
=> Caveat: This is an average case analysis of overlaps and best case analysis for number of clauses satisfied. This

Correction to Geometric distribution LLL estimate:
--------------------------------------------------
Previous bound assumes Pr[#SATclauses >= k] is geoemtric distribution. If k satisfying clauses are chosen from total

=> Pr[#SATclauses >= k] = summation(nCl*(1-p)^l*(p)^(n-l)), l=k,k+1,k+2,...,n

If k=n/2, this summation is exactly same as Condorcet Jury Theorem where each clause in the 3CNF is a voter = probabi

When k=n:
---------
Pr[#SATclauses = n] = (1-p)^n
From LLL, ep(d+1) > 1 => p > 1/e(d+1)
        1-p < 1 - 1/e(d+1)
        1-p < 1 - 0.36788/(d+1)
        (1-p)^n < (1 - 0.36788/(d+1))^n

d=0, no overlaps:
-----------------
        Pr[#SATclauses=n] = (1-p)^n < (0.6321)^n which tends to 0 for huge number of clauses n.

d=n-1, maximum overlaps:
------------------------
        Pr[#SATclauses=n] = (1-p)^n < (1 - 0.36788/n)^n
        But Limit(n->infinity) (1 - 0.36788/n)^n = e^(-0.36788) = 0.692200241
        Pr[#SATclauses=n] < 69.22% for huge n.

d=n/2, expected overlaps in uniform distribution:
-------------------------------------------------
        Pr[#SATclauses=n] = (1 - 1/(e(n/2+1)))^n = (1 - 0.735758882/(n+2))^n
        0.735758882/(n+2) < 0.735758882/n
        (1 - 0.735758882/(n+2))^n > (1 - 0.735758882/n)^n
        As n->infinity, (1-0.73578882/(n+2)) > e^(-0.73578882) ~ 47.91%
        => Pr[#SATclauses=n] > 47.91%

=> 97-98% satisfied clauses found in few hundred iterations above for upto 20 clauses and 20 variables should tend as
        Pr[#SATclauses >= 0.98n] = summation(nCl*(1-p)^l*(p)^(n-l)), l=k,k+1,k+2,...,n
This is tail bound for binomial distribution which is defined as:
        Pr[X >= k] <= e^(-nD(k/n || p))
where relative entropy D(k/n || p) = k/n log (k/np) + (1-k/n)log ((1-k/n)/(1-p))
But k/n = 0.98n/n = 0.98
=> D(k/n || p) = k/n log (k/np) + (1-k/n)log ((1-k/n)/(1-p)) = 0.98log(0.98/p) + 0.02log(0.02/(1-p))
=> Pr[X >= 0.98n] <= e^(-n*(0.98log(0.98/p) + 0.02log(0.02/(1-p))))
From LLL, p < 1/e(d+1). Substituting for p:
=> Pr[X >= 0.98n] <= e^(-n*(0.98log(0.98e(d+1)) + 0.02log(0.02(ed+e)/(ed+e-1))))
Lt(n->infinity) e^(-n*(0.98log(0.98e(d+1)) + 0.02log(0.02(ed+e)/(ed+e-1)))) = 0

Pr(#SATclauses <= n) <= e^(-(np-n)^2/2pn) by Chernoff bounds. From LLL, p < 1/e(d+1). For average number of overlaps
Pr(#SATclauses <= n) <= e^(-1/e) ~ 69.22% which coincides with maximum overlaps bound above.

Previous imply approximate CNF SAT solver by solving system of linear equations (= clauses relaxed to reals from bina

References:
-----------
450.1 Sum of Binomial Coefficients - http://web.maths.unsw.edu.au/~mikeh/webpapers/paper87.pdf - "... ???For years [t
George E. Andrews in 1973] I had been trying to point out that the rather confused world of binomial coefficient summ
450.2 Binomial Distribution Tail Bounds - https://en.wikipedia.org/wiki/Binomial_distribution#Tail_bounds

-------------------------------------------------------------------------------------------------------------------
451.(THEORY) Analysis of Error Upperbound for Approximate CNFSAT Solver based on Lovasz Local Lemma clause dependency
-------------------------------------------------------------------------------------------------------------------
(*) Least Squares lsqr() function has been replaced by recent lsmr() least square function mentioned to be faster.
(*) For 10 variables and 10 clauses for 54068 random 3CNF iterations following were the percentage of satisfied CNFs
per CNF:
Percentage of CNFs satisfied so far: 79.6722706172
Average Percentage of Clauses per CNF satisfied: 97.6276609517
```

(*) Again the percentage of clauses satisfied per random 3CNF converges to ~98% after ~54000 random 3CNFs.
(*) The clauses are created by randomly choosing each literal and its negation independent of any other literal or cl
(*) Repetitive convergence of number of clauses satisfied per CNF to 98% is mysterious in varied clause-variable comb
(*) Previous error bound analysis does not assume least squares and is based only on LLL. But Least Squares is about
(*) Motivation for least squares for boolean CNF is the relaxation achieved: An example assignment for a literal, 0.8
(*) Convergence to 98% implies probability of average number of clauses satisfied per 3CNF > 0.98n must be almost 0.
(*) Following Binomial Tail Bound:
Pr[*#SATclauses >= 0.98n] <= e^(-n*(0.98log(0.98e(d+1)) + 0.02log(0.02(ed+e)/(ed+e-1)))) is maximized when there are r*
Pr[*#SATclauses >= 0.98n] <= e^(-n*(0.98log(0.98e) + 0.02log(0.02(e)/(e-1))))*
Pr[*#SATclauses >= 0.98n] <= e^(-n*(0.41701 - 0.02999)*
Pr[*#SATclauses >= 0.98n] <= e^(-n*0.38701) ~ 0 for large n.*

452.(THEORY) PCP theorem, Hardness of Approximation, Least Squares SAT solver and Semidefinite Programming - related
and 3 September 2017

Probabilistically Checkable Proof theorem implies it is NP-hard to find satisfiable instances or 7/8+epsilon satisfia

References:

452.1 PCP and hardness of approximation - algorithm satisfying 7/8+delta clauses - https:*//cs.stackexchange.com/quest*
452.2 SDP for MAX3SAT - http:*//www.isa.ewi.tudelft.nl/~heule/publications/sum_of_squares.pdf - Maximum number of clau*
452.3 Binary Solutions to System of Linear Equations - https:*//arxiv.org/pdf/1101.3056.pdf*
452.4 Karloff-Zwick 7/8 algorithm for MAX3SAT - http:*//citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.51.1351 - A 7*
452.5 Inapproximability results - [Johan Hastad] - https:*//www.nada.kth.se/~johanh/optimalinap.pdf - Assuming P != NP*
452.6 Gauss-Jordan Elimination For System of Linear Equations, Origin of Least Squares, Linear Programs for Inequalit
452.7 Randomized Rounding for LP relaxation - https:*//en.wikipedia.org/wiki/Randomized_rounding - Randomized Rounding*

453.(THEORY) Approximate CNF 3SAT Solver - Least Squares Error Rounding Analysis - 9 September 2017,11 September 2017

Notation:
A' = Transpose of A
A^-1 = Inverse of A
E(A) = Expectation of A
p(A) = probability of A

Rounding in least squares solution to system of equations per CNF 3SAT fails if and only if binary assignment obtaine
        A'Ax = A'b
        => x = (A'A)^-1*A'*b
Number of satisfied clauses in least squares is = tr(A(A'A)^-1*A'*b) because Ax-b is minimized.
Maximum value of tr(A(A'A)^-1*A'*b) is the maximum number of satisfied clauses
if tr(A(A'A)^-1*A'*b) > 7/8*number_of_clauses, then P=NP by PCP theorem.

Each matrix A corresponding to a CNF is a random variable - A is a random matrix (not necessarily square).
Expected value of A(A'A)^-1*A'*b where random matrix A belongs to some probability distribution = E(A(A'A)^-1*A'*b)
        E(A(A'A)^-1*A'*b) = b*E(A)*E((A'A)^-1)*E(A')
        E(A(A'A)^-1*A'*b) = b*E(A)*E((A'A)^-1)*E(A) because A and A' are bijections.
        E(A(A'A)^-1*A'*b) = b*E(A)*E(A'A)*E(A) because A'A and (A'A)^-1 are bijections.
        E(A(A'A)^-1*A'*b) = b*E(A)*E(A)*E(A')*E(A) by definition of product of expectations
        E(A(A'A)^-1*A'*b) = b*(E(A))^4
        E(A(A'A)^-1*A'*b) = b*(A*p(A))^4
Each random matrix for a random 3CNF has nm entries where n is number of variables and m is number of clauses. Each c

Probability of choosing a variable or its negation = 1/(2n) for n variables + n negations.
Probability of a random matrix A of mn entries = (1/(2n))^nm
        => E(A(A'A)^-1*A'*b) = b(sigma(A(1/2n)^nm)^4 in uniform distribution.
                        = b(1/(2n)^nm * sigma(A))^4 which sums up all 2^nm possible boolean matrices.
                        = b(1/(2n)^mn * [matrix of 2^(nm-1) for all entries])^4
                        = b * [matrix of 2^(mn-1)/(2n)^nm for all entries]^4

==========
Tighter estimate for probability of a random 3CNF:
==========
        Per clause 3 literals have to be chosen from 2n literals = 2nC3(1/2n)^3(1/2n)^(2n-3)
        For m clauses each being independent, Probability of a random 3CNF
                = (2nC3(1/2n)^3(1/2n)^(2n-3))^m
==========
        E(A(A'A)^-1*A'*b) = b * [matrix of 2^(mn-1)/(2n)^nm for all entries]^4 (1,m) * (m*n, n*m, m*n, n*m)
                        = b * [matrix of n*2^(2nm-2)/(2n)^2mn for all entries]^2 (1,m) * (m*m, m*m)
                        = b * [matrix of m*n^2*2^(2nm-2)/(2n)^2mn for all entries] (1,m) * (m*m)
                        = [matrix of m^2*n^2*2^(2nm-2)/(2n)^2mn for all entries] (1,m)
        Trace of this expected matrix is the expected number of clauses that can be satisfied by least squares in uni
                        = m^3*n^2*2^(2nm-2)/(2n)^2mn
        if number of clauses to be satisfied <= m, m^3*n^2*2^(2nm-2) <= m*(2n)^2mn
                        => m^2*n^2*2^(2nm-2) <= (2n)^2nm
        which is obvious because (2n)^(2mn) grows faster.
                        => m^2*n^2*2^2nm <= 4 * 2^2nm * n^2nm
                        => m^2*n^2 <= 4 * n^2nm
        For all m clauses to be satisfied, this must be an equality:
                        m^2 = 4 * n^2(nm-1)

```
        if m=n (number of clauses = number of variables), all clauses are satisfied if:
                        n^2 = 4 * n^(n^2-1)
                        n^2/4 = n^(n^2-1)
                        => log (n^2/4) = (n^2-1) log(n)
                        => log (n^2/4)/log(n) = (n^2-1)
        which is a contradiction because RHS grows faster.
================================================
Alternatively, following simpler analysis leads to something more concrete:
        For minimum error x = (A'A)^-1*A'*b
        Ax=A(A'A)^-1*A'*b is maximum when A(A'A)^-1*A'*b = b
        => A(A'A)^-1*A' = I
When A is square symmetric matrix (number of variables = number of clauses ) and A=A':
        => A(AA)^-1*A = I
        => Least squares perfectly solves system of equations for CNFSAT if matrix of clauses is symmetric and square
Least squares thus solves a subset of NP-complete set of input 3CNFs i.e exactly solves a promise NP problem in deter

References:
-----------
453.1 Advanced Engineering Mathematics - [Erwin Kreyszig - 8th Edition] - Page 357
453.2 Linear Algebra - [Gilbert Strang] - http://math.mit.edu/~gs/linearalgebra/ila0403.pdf
453.3 Least Squares Error - [Stephen Boyd] - https://see.stanford.edu/materials/lsoeldsee263/05-ls.pdf

-------------------------------------------------------------------------------------------------------------------------
454. (THEORY) Random Matrix Analysis of Least Squares Approximate CNFSAT solver - related to 453 - 15 September 2017
-------------------------------------------------------------------------------------------------------------------------
From previous definition of expected value of random matrix equivalent to a random 3SAT:
       E(A(A'A)^-1*A'*b) = b*(E(A))^4
Previous derivation of expected value of a random matrix is further simplified by a different definition of expectati
       E(A) = [ matrix of E(x(i,j)) ] i.e each entry of the matrix is evaluated independently.
Let the probability of each entry x(i,j) of the random matrix = p. Each entry is a literal or its negation and can ta
       E(x(i,j)) = p
       => E(A) = [ matrix of p for all entries ]
       => b * (E(A))^4 = b*[ matrix of np^2 for all entries ]^2
                       = b*[ matrix of m*n^2*p^4 for all entries ]
                       = [ matrix of m^2*n^2*p^4 for all entries ]
       E(A(A'A)^-1*A'*b) = [ matrix of m^2*n^2*p^4 for all entries ]
       Trace(E(A(A'A)^-1*A'*b)) = m^3*n^2*p^4 which has maximum value equal to number of clauses m
       => m^3*n^2*p^4 <= m
       => m^2*n^2*p^4 <= 1
       => p^4 <= 1/(mn)^2
       => p^2 <= 1/(mn)
       => p <= 1/sqrt(mn)
This retrieves the probability distribution of the random 3SAT instances.
If all clauses have to be satisfied (ExactSAT), p = 1/sqrt(mn).
In previous example iterations of the solver, number of clauses = number of variables i.e p = 1/n.
This differs from the uniform probability assumption 1/(2n) per literal in earlier derivations including the negation
The promise subset of NP solved exactly by least squares is defined by the set of all random 3CNFs created with each
literal occurring with probability 1/n.

References:
----------
454.1 Random matrices - http://www.utstat.toronto.edu/~brunner/oldclass/431s09/readings/RandomMatrices.pdf

-------------------------------------------------------------------------------------------------------------------------
455. (FEATURE and THEORY) Commits - Approximate SAT solver - 15 September 2017
-------------------------------------------------------------------------------------------------------------------------
(*) Some debug statements have been included to print average probability of each literal's occurrence in random 3CNF
(*) SATsolver has been re-executed with 20 clauses and 20 variables and probability of occurrence of each literal has
(*) This was necessitated to ascertain the pseudorandomness of the clauses and CNFs created.
(*) Average probability of a literal is less than previous estimate of 1/n = 1/20 = 0.05 for 100% satisfied clauses b
(*) logs committed to testlogs/

-------------------------------------------------------------------------------------------------------------------------
456. (THEORY) Eigenvalues of Random Matrices, Riemann Zeta Function, MAXSAT ranking of merit, Promise problems and Sc
-------------------------------------------------------------------------------------------------------------------------
The set of random 3CNFs created by the solver is just a subset of all possible 3CNFs because all least squares API in

Earlier the problem of intrinsically ranking texts etc., based on merit has been reduced to a MAXSAT problem where ea

Random Matrix representation of random 3SAT instances throws up an unusual window into the realm of Riemann Zeta Func

References:
----------
456.1 Promise Problems - http://www.cs.tau.ac.il/~amnon/Classes/2017-BPP/Lectures/Lecture11.pdf
456.2 Survey of Promise Problems - [Oded Goldreich] - http://www.wisdom.weizmann.ac.il/~oded/PSX/prpr-r.pdf - there e
456.3 Gaussian Unitary Ensemble - Random Matrices - http://empslocal.ex.ac.uk/people/staff/mrwatkin//zeta/random.htm

-------------------------------------------------------------------------------------------------------------------------
457. (THEORY and FEATURE) CNF3SAT Approximate Solver Update - All possible variable-clause combinations - Commits - 1
-------------------------------------------------------------------------------------------------------------------------
(*) Solve_SAT2() function has been changed to take both number of variables and clauses as parameters
```

```
(*) Bug in EquationsB creation has been fixed so that length of b equals number of clauses. This was causing dimensic
dimensions in LSMR and LSQR
(*) Logs for 18*16 clauses-variables random 3CNFs have been committed to testlogs/. MaxSAT percentage is ~96% after 1
. Per literal probability is too less than 1/sqrt(mn) obtained from random matrix analysis and therefore converges mu
-----------------------------------
(*) Bugs fixed in prob_dist()
(*) restored lsmr()
(*) logs for some other clause-variable combinations have been committed to testlogs/. Observed per literal probabili
 matrix probability of 1/sqrt(mn)

-----------------------------------------------------------------------------------------------------------------------
458. (THEORY and FEATURE) Inapproximability and Random Matrix Analysis of Least Square Approximate SAT solver - Commi
-----------------------------------------------------------------------------------------------------------------------
Previous Random Matrix Analysis shows expected number of satisfied clauses = m^3*n^2*p^4 where there are m clauses,n
probability of choosing a positive or negative literal. For 21 variables and 20 clauses, following are the results af
-----------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------
Iteration : 29
---------------------------------------------------------------
solve_SAT2(): Verifying satisfying assignment computed .....
---------------------------------------------------------------
a: [[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0]
 [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]
 [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]
 [0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]
 [0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0]
 [1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0]]
b: [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
a.shape: (20, 21)
b.shape: (20,)
solve_SAT2(): lstsq(): x: (array([  5.00000000e-01,   0.00000000e+00,   0.00000000e+00,
         1.00000000e+00,   3.89920951e-13,   1.00000000e+00,
         1.00000000e+00,   5.00000000e-01,   7.50000000e-01,
         1.00000000e+00,   5.00000000e-01,   7.50000000e-01,
         1.24863661e-13,   0.00000000e+00,   0.00000000e+00,
         0.00000000e+00,   5.00000000e-01,   5.00000000e-01,
         1.00000000e+00,   5.00000000e-01,   0.00000000e+00]), 2, 12, 1.870828693386971, 5.2032984761241552e-12, 4.89
Random 3CNF: (!x11 + !x3 + !x13) * (!x2 + x12 + x11) * (x20 + x11 + !x10) * (!x1 + !x2 + x11) * (x5 + !x10 + x19) * (
Assignment computed from least squares: [1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0]
CNF Formula: [['!x11', '!x3', '!x13'], ['!x2', 'x12', 'x11'], ['x20', 'x11', '!x10'], ['!x1', '!x2', 'x11'], ['x5', '
Number of clauses satisfied: 20.0
Number of clauses : 20
Assignment satisfied: 1
Percentage of clauses satisfied: 100.0
Percentage of CNFs satisfied so far: 46.6666666667
Average Percentage of Clauses per CNF satisfied: 96.0
y= 46
sumfreq= 896
y= 43
sumfreq= 896
y= 36
sumfreq= 896
y= 46
sumfreq= 896
y= 41
sumfreq= 896
y= 42
sumfreq= 896
y= 48
sumfreq= 896
y= 37
sumfreq= 896
y= 48
sumfreq= 896
y= 39
sumfreq= 896
y= 46
```

```
sumfreq= 896
y= 42
sumfreq= 896
y= 42
sumfreq= 896
y= 50
sumfreq= 896
y= 39
sumfreq= 896
y= 33
sumfreq= 896
y= 41
sumfreq= 896
y= 42
sumfreq= 896
y= 44
sumfreq= 896
y= 43
sumfreq= 896
y= 48
sumfreq= 896
y= 40
sumfreq= 904
y= 43
sumfreq= 904
y= 51
sumfreq= 904
y= 36
sumfreq= 904
y= 38
sumfreq= 904
y= 46
sumfreq= 904
y= 44
sumfreq= 904
y= 36
sumfreq= 904
y= 41
sumfreq= 904
y= 45
sumfreq= 904
y= 46
sumfreq= 904
y= 34
sumfreq= 904
y= 46
sumfreq= 904
y= 46
sumfreq= 904
y= 36
sumfreq= 904
y= 42
sumfreq= 904
y= 44
sumfreq= 904
y= 47
sumfreq= 904
y= 51
sumfreq= 904
y= 54
sumfreq= 904
y= 38
sumfreq= 904
Probability of Variables chosen in CNFs so far: [0.05133928571428571, 0.04799107142857143, 0.04017857142857143, 0.051
Probability of Negations chosen in CNFs so far: [0.04424778761061947, 0.04756637168141593, 0.05641592920353982, 0.039
Average probability of a variable or negation: 0.047619047619
Probability per literal from Random Matrix Analysis of Least Squared (1/sqrt(mn)): 0.0487950036474
-----------------------------------------------------------------------------------------------------------
From PCP theorem and [Hastad] inapproximability result based on it, if m^3*n^2*p^4 > 7/8*m then P=NP.
=> m^2*n^2*p^4 > 7/8
=> p^4 > 7/(8m^2n^2)
=> if p > 0.96716821/sqrt(m*n) then P=NP.

In previous iteration, y and sumfreq print the frequencies of literals chosen so far at random (which are almost even
       Expected number of clauses satisfied = (20)^3*(21)^2*(0.047619)^4 = 18.1405 clauses or 90.7025%
implying the MaxSAT for 21 variables and 20 clauses converges to 90.7025% asymptotically ad infinitum.

The inequality m^3*n^2*p^4 > 7/8*m relates hardness and randomness because probability distribution for p is directly

For a binary valued random variable X, bias(X) is defined as Pr(X=0)-Pr(X=1) <= epsilon. Epsilon biased pseudorandom
```

Let Xi be the random variable for choosing the literal xi. If literal xi is chosen Xi=1, else Xi=0. Therefore bias(eP

An example application of Epsilon biased Pseudorandom Generator in reference 458.1 below :

=> bias(ePRG)=p-(1-p)=1.94/sqrt(m*n) - 1.
Equating this to the ePRG bias (n-1)/2^l:
=> (n-1)/2^l = [1.94/sqrt(m*n)] - 1
=> length of the seed l in ePRG = log([(n-1)*sqrt(m*n)]/[1.94-sqrt(m*n)]
Absurdity is for length of the seed to be valid logarithm, sqrt(m*n) < 1.94 => m*n < 4 which is possible only for m=1

$$\frac{2^{2l} * (1.94)^2}{n*[2^l + (n-1)]^2} < m$$

For large n, inequality tends to a surd 0 < m.

An example application of Epsilon biased Pseudorandom Generator in reference 458.2 below :

=> bias(ePRG)=p-(1-p)=1.94/sqrt(m*n) - 1.
Other ePRG (Mossel-Shpilka-Trevisan) bias 1/2^(kn/c^4) mentioned in the references relates m and n for random matrix

$$\frac{(1.94)^2 * 2^{(2kn/c^4)}}{n*(1 + 2^{(kn/c^4)})^2} < m$$

For large n, inequality tends to a surd 0 < m.

Both these prescribe a relation between n and m for small n and for large number of variables both ePRGs fit to the r

Generic Epsilon biased Pseudorandom Generator of bias epsilon=1/x for x > 0:

Consider a fictitious Epsilon biased Pseudorandom Generator of bias epsilon=1/x, x > 0.
Lowerbounding bias:

$$[1.94/\sqrt{m*n})] - 1 <= 1/x$$

which reduces to:

$$m >= (1.94)^2*x^2/[n*(x+1)^2]$$

This implies for any epsilon biased PRG, there exists number of clauses m below which Approximate SAT solver solves l

Number of possible random k-SAT instances of m clauses and n variables = (nP3 * 2)^m because each clause has either a

Number of possible 4 clause 3SATs are = (nP3 * 2)^4. Therefore fraction of random SAT instances not solvable for 7/8

$$\frac{(nP3 * 2)^4}{(nP3 * 2)^m} = \frac{1}{(n*(n-1)*(n-2))^{(m-4)}}$$

which is negligible for large m and n but non-trivial for small m and n.

Number of variables in 4 clause 3SAT is at the maximum 12. Thus this small subset can be solved in a constant time be

[Caution: Previous derivation is still experimental with possible errors and assumes an epsilon PRG of bias atleast [

References:

458.1 Epsilon biased Pseudo Random Generators - [Advanced Complexity Theory Course Notes - Dieter] - http://pages.cs.
458.2 Epsilon biased Pseudo Random Generator in NC0 - every pseudorandom bit depends only on 5 bits of seed - [Elchan

---
459. (THEORY) Intrinsic Merit, Consensus Algorithms, Byzantine Failures and Level Playing Field - 23 September 2017

So far Intrinsic merit of a text has been analyzed mostly in the context of connectedness and meaningfulness of it. I
* Agreement - all parties must agree on a correct process and its outcome
* Validity - if all correct processes receive same input value,they must all output same value
* Termination - all processes must eventually decide on an output value
There are realworld consensus implementations - Google Chubby Lock Service based on Paxos Consensus Protocol, Bitcoin

Consensus can be impeded by presence of malicious nodes which spoof and send spurious votes known as Byzantine genera

References:

459.1 Consensus Algorithms - https://en.wikipedia.org/wiki/Consensus_(computer_science)

---
460. (THEORY) Random Matrix Rounding for Least Squares Approximate CNFSAT Solver, Distinguisher for Pseudorandomness,
- related to 318, 458 - 13 October 2017

Existence of Pseudorandom generators (PRG) implies P != NP. Proof of this is by contradiction: If there exists a dist

Majority Voting Hardness Lemma is an adaptation of Yao's XOR Lemma for hardness amplification from weak voting functi

Random Matrix Rounding for Least Squares Approximate CNFSAT Solver in previous sections derives an expected probabili

Probability of choosing a CNF by RMLSQR (there are 3m literals per 3CNF) = (1/[mn])^1.5m

Alternatively,number of all possible random 3CNF SATs of m clauses and n variables = (n * n * n)^m = n^(3m)

Probability of choosing a CNF from all possible n^(3m) random 3SATs in this uniform distribution is = (1/n)^3m

```
Thus there are two possible probability distributions for choosing a random 3SAT: (1/[mn])^1.5m required by Random Ma

If m=n(number of clauses and number of variables are equal):
-------------------------------------------------------------
(1/[nn])^1.5n = (1/n)^3n and thus both RMLSQR and Uniform distributions are same implying similar pseudorandomness in

If m !=n e.g m >> n (this is most prevalent setting where number of clauses are huge and variables are relatively les
-----------------------------------------------------------------------------------------------------------------------
(1/[mn])^1.5m < (1/n)^3m
(1/m)^1.5m * (1/n)^1.5m < (1/n)^1.5m * (1/n)^1.5m
(1/m)^1.5m < (1/n)^1.5m
1/m < 1/n
=> m > n

When number of clauses m differs from number of variables n, RMLSQR and Uniform distributions are dissimilar implying
randomness-es: PRG1 for RMLSQR and PRG2(or perfect randomness) for Uniform. Distinguisher for these two random genera

-----------------------------------------------------------------------------------------------------------------------
461. (THEORY and FEATURE) Random Matrix Rounding for Least Squares Approximate CNFSAT Solver - various clause-variabl
-----------------------------------------------------------------------------------------------------------------------
Following are some random 3SAT iteration MAXSAT percentage numbers for multiple combinations of number of variables a
this could be error in estimating linux PRNG probability distribution. Deficiencies of Linux PRNGs - especially randc

##############################################
17 variables, 18 clauses - 1066 random 3SATs:
##############################################
-----------------------------------------------------------------
Iteration : 1066
-----------------------------------------------------------------
solve_SAT2(): Verifying satisfying assignment computed .....
-----------------------------------------------------------------
a: [[0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0]
 [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0]
 [0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]
 [0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0]
 [0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0]
 [0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0]]
b: [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
a.shape: (18, 17)
b.shape: (18,)
solve_SAT2(): lstsq(): x: (array([  0.00000000e+00,   1.00000000e+00,   8.00000000e-01,
         1.00000000e+00,   2.00000000e-01,   1.00000000e+00,
         1.00000000e+00,   6.66666667e-01,   6.00000000e-01,
         0.00000000e+00,   0.00000000e+00,  -6.66666667e-01,
         6.00000000e-01,   1.89190298e-14,   6.66666667e-01,
         4.00000000e-01,   0.00000000e+00]), 2, 11, 1.653279569018299, 9.190906242470613e-13, 4.690415759823429, 4.19
Random 3CNF: (x3 + !x11 + !x16) * (!x16 + x3 + x5) * (x6 + !x9 + !x4) * (x15 + !x10 + x8) * (!x14 + x2 + !x8) * (!x11
Assignment computed from least squares: [0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0]
CNF Formula: [['x3', '!x11', '!x16'], ['!x16', 'x3', 'x5'], ['x6', '!x9', '!x4'], ['x15', '!x10', 'x8'], ['!x14', 'x2
Number of clauses satisfied: 18.0
Number of clauses : 18
Assignment satisfied: 1
Percentage of clauses satisfied: 100.0
Percentage of CNFs satisfied so far: 61.0121836926
Average Percentage of Clauses per CNF satisfied: 97.3341664063
y= 1675
sumfreq= 28722
y= 1683
sumfreq= 28722
y= 1716
sumfreq= 28722
y= 1750
sumfreq= 28722
y= 1699
sumfreq= 28722
y= 1690
sumfreq= 28722
y= 1697
```

```
sumfreq= 28722
y= 1662
sumfreq= 28722
y= 1732
sumfreq= 28722
y= 1740
sumfreq= 28722
y= 1687
sumfreq= 28722
y= 1669
sumfreq= 28722
y= 1677
sumfreq= 28722
y= 1662
sumfreq= 28722
y= 1664
sumfreq= 28722
y= 1660
sumfreq= 28722
y= 1659
sumfreq= 28722
y= 1716
sumfreq= 28896
y= 1782
sumfreq= 28896
y= 1660
sumfreq= 28896
y= 1714
sumfreq= 28896
y= 1704
sumfreq= 28896
y= 1716
sumfreq= 28896
y= 1721
sumfreq= 28896
y= 1688
sumfreq= 28896
y= 1679
sumfreq= 28896
y= 1653
sumfreq= 28896
y= 1662
sumfreq= 28896
y= 1686
sumfreq= 28896
y= 1702
sumfreq= 28896
y= 1673
sumfreq= 28896
y= 1735
sumfreq= 28896
y= 1678
sumfreq= 28896
y= 1727
sumfreq= 28896
Probability of Variables chosen in CNFs so far: [0.058317665900703294, 0.05859619803634844, 0.05974514309588469, 0.06
Probability of Negations chosen in CNFs so far: [0.059385382059800665, 0.061669435215946845, 0.05744739756367663, 0.0
Average probability of a variable or negation: 0.0588235294118
Probability per literal from Random Matrix Analysis of Least Squared (1/sqrt(mn)): 0.0571661950475

##########################################
18 variables, 19 clauses - 137 random 3SATs:
##########################################
------------------------------------------------------------
Iteration : 137
------------------------------------------------------------
solve_SAT2(): Verifying satisfying assignment computed .....
------------------------------------------------------------
a: [[1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0]
 [0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0]
 [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1]
 [0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0]
 [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0]
 [0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 1 0 0]
```

```
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0]
 [0 0 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]]
b: [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
a.shape: (19, 18)
b.shape: (19,)
solve_SAT2(): lstsq(): x: (array([  6.66666667e-01,   1.00000000e+00,  -1.11111111e-01,
         3.33333333e-01,   0.00000000e+00,   1.00000000e+00,
         4.44444445e-01,   1.66666667e+00,   8.93039852e-11,
         0.00000000e+00,   0.00000000e+00,   6.66666667e-01,
        -6.66666666e-01,   0.00000000e+00,  -5.55555556e-01,
         6.66666667e-01,  -8.43769499e-15,   1.00000000e+00]), 2, 13, 2.0816659994661344, 5.027196399901854e-09, 5.29
Random 3CNF: (x4 + x1 + !x7) * (!x3 + x6 + x9) * (x3 + x12 + x7) * (!x13 + !x4 + !x10) * (x12 + x16 + !x6) * (x12 + !
Assignment computed from least squares: [1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1]
CNF Formula: [['x4', 'x1', '!x7'], ['!x3', 'x6', 'x9'], ['x3', 'x12', 'x7'], ['!x13', '!x4', '!x10'], ['x12', 'x16',
Number of clauses satisfied: 19.0
Number of clauses : 19
Assignment satisfied: 1
Percentage of clauses satisfied: 100.0
Percentage of CNFs satisfied so far: 52.8985507246
Average Percentage of Clauses per CNF satisfied: 96.7581998474
y= 220
sumfreq= 3952
y= 243
sumfreq= 3952
y= 252
sumfreq= 3952
y= 230
sumfreq= 3952
y= 195
sumfreq= 3952
y= 203
sumfreq= 3952
y= 231
sumfreq= 3952
y= 225
sumfreq= 3952
y= 224
sumfreq= 3952
y= 228
sumfreq= 3952
y= 221
sumfreq= 3952
y= 224
sumfreq= 3952
y= 205
sumfreq= 3952
y= 216
sumfreq= 3952
y= 199
sumfreq= 3952
y= 206
sumfreq= 3952
y= 203
sumfreq= 3952
y= 227
sumfreq= 3952
y= 207
sumfreq= 3914
y= 218
sumfreq= 3914
y= 214
sumfreq= 3914
y= 220
sumfreq= 3914
y= 223
sumfreq= 3914
y= 204
sumfreq= 3914
y= 220
sumfreq= 3914
y= 233
sumfreq= 3914
y= 203
sumfreq= 3914
y= 210
sumfreq= 3914
y= 234
```

```
sumfreq= 3914
y= 208
sumfreq= 3914
y= 239
sumfreq= 3914
y= 180
sumfreq= 3914
y= 205
sumfreq= 3914
y= 232
sumfreq= 3914
y= 259
sumfreq= 3914
y= 205
sumfreq= 3914
Probability of Variables chosen in CNFs so far: [0.05566801619433198, 0.06148785425101214, 0.06376518218623482, 0.058
Probability of Negations chosen in CNFs so far: [0.05288707204905468, 0.05569749616760347, 0.054675523760858456, 0.05
Average probability of a variable or negation: 0.0555555555556
Probability per literal from Random Matrix Analysis of Least Squared (1/sqrt(mn)): 0.0540738070436

############################################################
19 variables, 18 clauses - 34 random 3SATs
############################################################
------------------------------------------------------------
Iteration : 34
------------------------------------------------------------
solve_SAT2(): Verifying satisfying assignment computed .....
------------------------------------------------------------
a: [[0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0]
 [0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0]
 [0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]
 [0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]
 [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0]
 [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1]
 [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
b: [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
a.shape: (18, 19)
b.shape: (18,)
solve_SAT2(): lstsq(): x: (array([  1.00000000e+00,   1.00000000e+00,   1.00000000e+00,
         1.00000000e+00,   1.00000000e+00,   1.00000000e+00,
        -3.65332764e-15,   0.00000000e+00,   0.00000000e+00,
         9.05525654e-15,   1.00000000e+00,   1.00000000e+00,
         0.00000000e+00,   9.05525654e-15,   0.00000000e+00,
        -1.00000000e+00,   1.00000000e+00,   3.34888367e-14,
        -3.65332764e-15]), 2, 12, 1.414213562373095, 2.093877713811129e-13, 4.795831523312719, 3.8125525032467102, 3.
Random 3CNF: (x4 + x14 + x10) * (!x2 + !x11 + x5) * (x16 + x2 + x17) * (!x8 + x11 + x7) * (!x5 + x11 + !x14) * (x4 +
Assignment computed from least squares: [1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0]
CNF Formula: [['x4', 'x14', 'x10'], ['!x2', '!x11', 'x5'], ['x16', 'x2', 'x17'], ['!x8', 'x11', 'x7'], ['!x5', 'x11',
Number of clauses satisfied: 18.0
Number of clauses : 18
Assignment satisfied: 1
Percentage of clauses satisfied: 100.0
Percentage of CNFs satisfied so far: 80.0
Average Percentage of Clauses per CNF satisfied: 98.7301587302
y= 52
sumfreq= 942
y= 39
sumfreq= 942
y= 46
sumfreq= 942
y= 50
sumfreq= 942
y= 53
sumfreq= 942
y= 43
sumfreq= 942
y= 47
sumfreq= 942
y= 51
sumfreq= 942
```

```
y= 57
sumfreq= 942
y= 48
sumfreq= 942
y= 54
sumfreq= 942
y= 49
sumfreq= 942
y= 46
sumfreq= 942
y= 61
sumfreq= 942
y= 54
sumfreq= 942
y= 44
sumfreq= 942
y= 48
sumfreq= 942
y= 49
sumfreq= 942
y= 51
sumfreq= 942
y= 44
sumfreq= 948
y= 42
sumfreq= 948
y= 43
sumfreq= 948
y= 43
sumfreq= 948
y= 61
sumfreq= 948
y= 60
sumfreq= 948
y= 49
sumfreq= 948
y= 60
sumfreq= 948
y= 43
sumfreq= 948
y= 57
sumfreq= 948
y= 47
sumfreq= 948
y= 60
sumfreq= 948
y= 49
sumfreq= 948
y= 41
sumfreq= 948
y= 60
sumfreq= 948
y= 46
sumfreq= 948
y= 53
sumfreq= 948
y= 41
sumfreq= 948
y= 49
sumfreq= 948
Probability of Variables chosen in CNFs so far: [0.055201698513800426, 0.041401273885350316, 0.04883227176220807, 0.0
Probability of Negations chosen in CNFs so far: [0.046413502109704644, 0.04430379746835443, 0.04535864978902954, 0.04
Observed Average probability of a variable or negation: 0.0526315789474
Probability per literal from Random Matrix Analysis of Least Squared (1/sqrt(mn)): 0.0540738070436
Percentage of Clauses satisfied - Observed Average Probability substituted in Random Matrix Analysis of Least Squared

############################################################
16 variables, 15 clauses - 60 random 3SATs
############################################################
------------------------------------------------------------
Iteration : 60
------------------------------------------------------------
solve_SAT2(): Verifying satisfying assignment computed .....
------------------------------------------------------------
a: [[0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0]
 [0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0]
```

```
  [0 1 0 0 1 0 0 0 0 1 0 0 0 0 0 0]
  [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
  [0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0]
  [0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0]
  [0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0]
  [0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0]
  [0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0]
  [0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0]]
b: [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
a.shape: (15, 16)
b.shape: (15,)
solve_SAT2(): lstsq(): x: (array([  0.00000000e+00,   4.00000000e-01,   1.00000000e+00,
         1.00000000e+00,  -2.00000000e-01,   1.00000000e+00,
         0.00000000e+00,   6.00000000e-01,   6.00000000e-01,
         8.00000000e-01,  -2.28234722e-16,   0.00000000e+00,
         5.00000000e-01,   5.00000000e-01,   1.00000000e+00,
         0.00000000e+00]), 2, 9, 1.5491933384829675, 1.3065051242742054e-12, 4.242640687119285, 2.7186789946524619, 2
Random 3CNF: (!x8 + x10 + !x11) * (x11 + x3 + !x8) * (!x10 + !x16 + !x1) * (!x8 + x6 + !x4) * (x9 + !x1 + !x10) * (!x
Assignment computed from least squares: [0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0]
CNF Formula: [['!x8', 'x10', '!x11'], ['x11', 'x3', '!x8'], ['!x10', '!x16', '!x1'], ['!x8', 'x6', '!x4'], ['x9', '!x
Number of clauses satisfied: 15.0
Number of clauses : 15
Assignment satisfied: 1
Percentage of clauses satisfied: 100.0
Percentage of CNFs satisfied so far: 65.5737704918
Average Percentage of Clauses per CNF satisfied: 97.1584699454
y= 75
sumfreq= 1376
y= 95
sumfreq= 1376
y= 77
sumfreq= 1376
y= 79
sumfreq= 1376
y= 96
sumfreq= 1376
y= 89
sumfreq= 1376
y= 79
sumfreq= 1376
y= 94
sumfreq= 1376
y= 86
sumfreq= 1376
y= 93
sumfreq= 1376
y= 101
sumfreq= 1376
y= 91
sumfreq= 1376
y= 92
sumfreq= 1376
y= 80
sumfreq= 1376
y= 77
sumfreq= 1376
y= 72
sumfreq= 1376
y= 93
sumfreq= 1369
y= 82
sumfreq= 1369
y= 82
sumfreq= 1369
y= 85
sumfreq= 1369
y= 84
sumfreq= 1369
y= 87
sumfreq= 1369
y= 84
sumfreq= 1369
y= 71
sumfreq= 1369
y= 80
sumfreq= 1369
y= 74
sumfreq= 1369
y= 90
sumfreq= 1369
y= 91
```

```
sumfreq= 1369
y= 100
sumfreq= 1369
y= 100
sumfreq= 1369
y= 88
sumfreq= 1369
y= 78
sumfreq= 1369
Probability of Variables chosen in CNFs so far: [0.05450581395348837, 0.0690406976744186, 0.0559593023255814, 0.0574
Probability of Negations chosen in CNFs so far: [0.0679327976625274, 0.05989773557341125, 0.05989773557341125, 0.062
Observed Average probability of a variable or negation: 0.0625
Probability per literal from Random Matrix Analysis of Least Squared (1/sqrt(mn)): 0.0645497224368
Percentage of Clauses satisfied - Observed Average Probability substituted in Random Matrix Analysis of Least Squared

##################################################
21 variables, 18 clauses - 6 random 3SATs
##################################################
------------------------------------------------------------
Iteration : 6
------------------------------------------------------------
solve_SAT2(): Verifying satisfying assignment computed .....
------------------------------------------------------------
a: [[0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 1 0 0 0 0 0]
 [0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0]
 [0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
 [0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0]
 [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0]
 [0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0]
 [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
b: [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
a.shape: (18, 21)
b.shape: (18,)
solve_SAT2(): lstsq(): x: (array([  5.00000000e-01,   1.00000000e+00,   0.00000000e+00,
         5.00000000e-01,   1.00000000e+00,   0.00000000e+00,
        -2.63027447e-16,   1.00000000e+00,  -3.67351548e-14,
         0.00000000e+00,   8.39171896e-12,   9.28077060e-17,
         0.00000000e+00,  -3.67351548e-14,   0.00000000e+00,
         2.65906351e-12,   1.00000000e+00,  -2.52681990e-14,
         5.00000000e-01,   1.00000000e+00,   5.00000000e-01]), 2, 12, 1.7320508075688767, 6.3185165416969518e-11, 4.8
Random 3CNF: (x11 + x8 + x16) * (!x12 + x11 + x5) * (!x4 + x17 + x18) * (x5 + !x18 + !x8) * (!x8 + x21 + x1) * (!x9 +
Assignment computed from least squares: [1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1]
CNF Formula: [['x11', 'x8', 'x16'], ['!x12', 'x11', 'x5'], ['!x4', 'x17', 'x18'], ['x5', '!x18', '!x8'], ['!x8', 'x21
Number of clauses satisfied: 18.0
Number of clauses : 18
Assignment satisfied: 1
Percentage of clauses satisfied: 100.0
Percentage of CNFs satisfied so far: 71.4285714286
Average Percentage of Clauses per CNF satisfied: 97.619047619
y= 10
sumfreq= 184
y= 9
sumfreq= 184
y= 5
sumfreq= 184
y= 7
sumfreq= 184
y= 10
sumfreq= 184
y= 2
sumfreq= 184
y= 11
sumfreq= 184
y= 5
sumfreq= 184
y= 8
sumfreq= 184
y= 13
sumfreq= 184
y= 14
```

```
sumfreq= 184
y= 9
sumfreq= 184
y= 8
sumfreq= 184
y= 10
sumfreq= 184
y= 4
sumfreq= 184
y= 9
sumfreq= 184
y= 11
sumfreq= 184
y= 9
sumfreq= 184
y= 10
sumfreq= 184
y= 9
sumfreq= 184
y= 11
sumfreq= 184
y= 8
sumfreq= 194
y= 14
sumfreq= 194
y= 9
sumfreq= 194
y= 11
sumfreq= 194
y= 7
sumfreq= 194
y= 10
sumfreq= 194
y= 11
sumfreq= 194
y= 11
sumfreq= 194
y= 11
sumfreq= 194
y= 11
sumfreq= 194
y= 6
sumfreq= 194
y= 11
sumfreq= 194
y= 4
sumfreq= 194
y= 6
sumfreq= 194
y= 11
sumfreq= 194
y= 3
sumfreq= 194
y= 8
sumfreq= 194
y= 14
sumfreq= 194
y= 4
sumfreq= 194
y= 13
sumfreq= 194
y= 11
sumfreq= 194
Probability of Variables chosen in CNFs so far: [0.05434782608695652, 0.04891304347826087, 0.02717391304347826, 0.038
Probability of Negations chosen in CNFs so far: [0.041237113402061855, 0.07216494845360824, 0.04639175257731959, 0.05
Observed Average probability of a variable or negation: 0.047619047619
Probability per literal from Random Matrix Analysis of Least Squared (1/sqrt(mn)): 0.0514344499874
Percentage of Clauses satisfied - Observed Average Probability substituted in Random Matrix Analysis of Least Squared

-----------------------------------------------------------------------------------------------------------------------
462. (THEORY) Space-Filling, Random Close Packing, Bin Packing and Voter Decision Functions - 27 October 2017 - relat
-----------------------------------------------------------------------------------------------------------------------
Parallel PRG and Cellular Automaton Algorithms for randomly filling a space with objects and their relevance to Linea

Reference:
---------
462.1 Random Close Packings of Spheres in a Container - Space-filling and Structural Topology - http://www.math.corne

-----------------------------------------------------------------------------------------------------------------------
463. (THEORY) Reduction from Random Close Packing to CNFSAT - related to 462 - 30 October 2017
-----------------------------------------------------------------------------------------------------------------------
```

```
Each random close pack after a random shuffle shifts the centre of an n-sphere. Set of all possible centroids of an n
       (C11 V C12 V C13 ...) /\ (C21 V C22 V C32 ...) /\ (C31 V C32 V C33 ...) /\ ...
Each Cik is assumed to be boolean variable which is 1 if n-sphere si is located in centroid Cik in random close pack

Randomized Algorithm (Parallel PRG or Cellular Automaton) for this space filling random close pack finds a satisfying
pack is the limit on density = Total Volume of Spheres (or) Total number of satisfied clauses / Volume of Container.

Filling the space within the container by n-spheres in parallel monte carlo random choice of centroids, simulates man

-----------------------------------------------------------------------------------------------------------------
464. (FEATURE-DONE) Ephemeris Search Script Update - Celestial Pattern Mining - 31 October 2017
-----------------------------------------------------------------------------------------------------------------
Updated Ephemeris Search for Sequence Mined Celestial Configurations - toString() function has been changed to concat
for vacant zodiac signs while creating encoded celestial chart.

-----------------------------------------------------------------------------------------
465. (FEATURE-DONE) Ephemeris Search - Sequence Mining of Tropical Monsoon for mid-November 2017 - 1 November 2017
-----------------------------------------------------------------------------------------
(#) Apriori GSP SequenceMining on autogen_classifier_dataset historic Storms data has been executed
(#) asfer.enchoros.seqmining has been updated from autogen_classifier_dataset
(#) MinedClassAssociationRules.txt has been rewritten containing almost 2500 celestial planetary patterns
(#) Ephemeris has been searched for almost 250 top astronomical patterns of these 2500 configurations
for mid-November 2017
(#) There has been a significant match of most the patterns during this period. Range of search has been
narrowed to 2 days because exhaustive search for all patterns is intensive.
(#) This matches to NOAA CPC forecast in http://www.cpc.ncep.noaa.gov/products/JAWF_Monitoring/India/GFS_forecasts.sh

-----------------------------------------------------------------------------------------
466. (THEORY) Discrete Hyperbolic Computational Geometric Factorization, Chvatal Art Gallery Theorem
 and Parallel Tiling - 6 November 2017, 8 November 2017 - related to 34
-----------------------------------------------------------------------------------------
Chvatal Art Gallery Theorem states for floodlighting an art gallery shaped as n-polygon, floor(n/3) guards are suffic

As opposed to polygon pixelation, if the hyperbolic curve is discretized into set of line segments by plain rounding

References:
----------
466.1 Chvatal Art Gallery Theorem and Fisk's Proof - https://www.cut-the-knot.org/Curriculum/Combinatorics/Chvatal.sh
466.2 Number of prime factors of an integer - [Srinivasa Ramanujan] - Quarterly Journal of Mathematics, XLVIII, 1917,
466.3 Introduction to Parallel Algorithms - [Joseph JaJa] - Chapter 4 - Searching,Sorting,Merging - Corollary 4.5 - C

-----------------------------------------------------------------------------------------
467. (FEATURE-DONE) Support Vector Machines implementation - based on CVXPY - 9 November 2017
-----------------------------------------------------------------------------------------
(*) New Support Vector Machines python implementation is committed to NeuronRain AsFer repository
(*) This minimizes an objective function 1/2*||w|| subject to constraint ||WX+b|| >= 1 which
labels a point +1 or -1 on either side of a decision separating hyperplane WX+b for weight vector W
and bias b (Reference: Machine Learning - Ethem Alpaydin - Chapter 13 - Kernel Machines)
(*) Optimization is solved by CVXPY DCCP Convex-Concave program
(*) logs for this have been added to testlogs/
(*) Present NeuronRain AsFer SVMRetriever.cpp depends on third-party SVMlight opensource software. With
 this new implementation, references to SVMlight are to be phased out.

-----------------------------------------------------------------------------------------------------------------
468. (THEORY) Random Matrix Rounding for CNFSAT Solver, Blum-Micali PRG, Distinguisher for Pseudorandomness - 10 Nove
related to 460
-----------------------------------------------------------------------------------------------------------------
Blum-Micali PRG depends on intractability of Discrete Logarithm f(x)=g^x mod p for primes p,g and group element x. Bl
sequences of (f(x), f^2(x), f^3(x), ...) by composition of f and computes stream of bits by hard-core boolean predica

Probability of choosing a CNF by RMLSQR probability (p=1/sqrt(mn) and there are 3m literals per 3CNF) = (1/[mn])^1.5m
Probability of choosing a CNF from all possible n^(3m) random 3SATs in uniform distribution is = (1/n)^3m

Inverse of RMLSQR probability = (mn)^1.5m is the expected number of pseudorandom binary strings churned out by PRG be
Inverse of uniform probability = n^3m is the expected number of pseudorandom binary strings churned out by PRG before

Distinguisher for these 2 distributions iterates through both sequences of bitstreams and prints "RMLSQR" if match oc

From PCP [Hastad] inapproximability result and previous Random Matrix analysis for approximate CNFSATSolver, if m^3*n

References:
-----------
468.1 Existence of Pseudorandom Generators - [Goldreich-Hugo-Luby] - http://www.wisdom.weizmann.ac.il/~oded/X/gkl.pdf

-----------------------------------------------------------------------------------------
469. (FEATURE-DONE) Support Vector Machines - update - 10 November 2017
-----------------------------------------------------------------------------------------
(*) Numpy indexing has been changed
(*) Both random point and parametric point distances have been tested
(*) DCCP log and Support Vector logs for 2 points and a random point have been committed to testlogs/
(*) Distances are printed in the matrix result - two diametrically opposite points have equal distances
```

```
(*) Distance matrix is returned from distance_from_separating_hyperplane() function
(*) The distance minimized is the L1 norm (sum of tuple elements) and not L2 norm
 (sum of squares of tuple elements)

---------------------------------------------------------------------------------------------------------------------
470. (FEATURE-DONE) Computational Geometric Hyperbolic Factorization - Pixelated Segments Spark Bitonic Sort - Single
 Benchmarks - 13 November 2017
---------------------------------------------------------------------------------------------------------------------
(*) Numbers 147,219,251,253 are factorized.
(*) C++ tiling pre-processing routines have been changed for this and Pixelated Tiles storage text files for bitonic
(*) Factorization is benchmarked on single node cluster on dual core (which is parallel RAM).
(*) This is just a representative number on single dual-core CPU and not a cloud parallelism benchmark.
(*) Each DAGScheduler Spark work item is independent code executable in parallel and benchmark has to be this paralle
is captured in per task duration logs below by Spark Driver:
17/11/13 21:18:22 INFO TaskSetManager: Finished task 1.0 in stage 1791.0 (TID 3583) in 46 ms on localhost (executor d
17/11/13 21:18:22 INFO TaskSetManager: Finished task 0.0 in stage 1791.0 (TID 3582) in 47 ms on localhost (executor d
17/11/13 21:18:22 INFO TaskSetManager: Starting task 0.0 in stage 1792.0 (TID 3584, localhost, executor driver, parti
17/11/13 21:18:22 INFO TaskSetManager: Starting task 1.0 in stage 1792.0 (TID 3585, localhost, executor driver, parti
17/11/13 21:18:22 INFO TaskSetManager: Finished task 1.0 in stage 1792.0 (TID 3585) in 46 ms on localhost (executor d
17/11/13 21:18:22 INFO TaskSetManager: Finished task 0.0 in stage 1792.0 (TID 3584) in 50 ms on localhost (executor d
17/11/13 21:26:39 INFO TaskSetManager: Finished task 0.0 in stage 1791.0 (TID 3582) in 50 ms on localhost (executor d
17/11/13 21:26:39 INFO TaskSetManager: Finished task 1.0 in stage 1791.0 (TID 3583) in 49 ms on localhost (executor d
17/11/13 21:26:39 INFO TaskSetManager: Starting task 0.0 in stage 1792.0 (TID 3584, localhost, executor driver, parti
17/11/13 21:26:39 INFO TaskSetManager: Starting task 1.0 in stage 1792.0 (TID 3585, localhost, executor driver, parti
17/11/13 21:26:39 INFO TaskSetManager: Finished task 0.0 in stage 1792.0 (TID 3584) in 48 ms on localhost (executor d
17/11/13 21:26:39 INFO TaskSetManager: Finished task 1.0 in stage 1792.0 (TID 3585) in 49 ms on localhost (executor d
17/11/13 21:33:59 INFO TaskSetManager: Finished task 0.0 in stage 1791.0 (TID 3582) in 48 ms on localhost (executor d
17/11/13 21:33:59 INFO TaskSetManager: Finished task 1.0 in stage 1791.0 (TID 3583) in 50 ms on localhost (executor d
17/11/13 21:33:59 INFO TaskSetManager: Starting task 0.0 in stage 1792.0 (TID 3584, localhost, executor driver, parti
17/11/13 21:33:59 INFO TaskSetManager: Starting task 1.0 in stage 1792.0 (TID 3585, localhost, executor driver, parti
17/11/13 21:33:59 INFO TaskSetManager: Finished task 0.0 in stage 1792.0 (TID 3584) in 55 ms on localhost (executor d
17/11/13 21:33:59 INFO TaskSetManager: Finished task 1.0 in stage 1792.0 (TID 3585) in 57 ms on localhost (executor d
17/11/13 21:40:20 INFO TaskSetManager: Finished task 1.0 in stage 1791.0 (TID 3583) in 49 ms on localhost (executor d
17/11/13 21:40:20 INFO TaskSetManager: Finished task 0.0 in stage 1791.0 (TID 3582) in 52 ms on localhost (executor d
17/11/13 21:40:20 INFO TaskSetManager: Starting task 0.0 in stage 1792.0 (TID 3584, localhost, executor driver, parti
17/11/13 21:40:20 INFO TaskSetManager: Starting task 1.0 in stage 1792.0 (TID 3585, localhost, executor driver, parti
17/11/13 21:40:20 INFO TaskSetManager: Finished task 0.0 in stage 1792.0 (TID 3584) in 50 ms on localhost (executor d
17/11/13 21:40:20 INFO TaskSetManager: Finished task 1.0 in stage 1792.0 (TID 3585) in 50 ms on localhost (executor d

(*) Bitonic Sort is O(logN*logN) and number of processors required is approximately O(N^2.5) but abides by NC definit

---------------------------------------------------------------------------------------------------------------------
471. (FEATURE-DONE) Support Vector Machines Update - Learn and Classify functions - 13 November 2017
---------------------------------------------------------------------------------------------------------------------
(*) Support Vector Machines python implementation has been updated to include two functions :
        - for learning set of support vectors from a training dataset tuples and store the vectors in a dictionary ma
        - to classify a tuple by finding its distance with reference to the support vector regions - distance-to-vect
        - if there are more than 1 tuples per minimum distance, those have to be reckoned as support vectors

---------------------------------------------------------------------------------------------------------------------
472. (THEORY) Jones-Sato-Wada-Wiens Theorem, Complement Functions, Prime-Composite Complementation, Prime Number Theo
Ramsey 2-coloring of integers, Hilbert Tenth Problem, Unique Factorization , Matiyasevich-Robinson-Davis-Putnam Theor
related to 24,370,390,394 - 13 November 2017, 14 November 2017, 29 November 2017,1 December 2017,4 December 2017,23 D
(Draft updates to https://arxiv.org/abs/1106.4102)
---------------------------------------------------------------------------------------------------------------------
Jones-Sato-Wada-Wiens Theorem proves existence of a polynomial in 25 degree-26 variables which has values equal to se

Number of primes < p=xy:
        = c1*xy/logxy
Number of primes < q=(x+1)y:
        = c1*(x+1)y/log(x+1)y

Number of primes between p=xy and q=(x+1)y :
        = c1*(x+1)y/log(x+1)y - c1*xy/logxy

Assuming number of primes between p and q = c1*(x+1)y/log(x+1)y - c1*xy/logxy > 0:
        (x+1)y/log(x+1)y > xy/logxy
After reducing:
        log(xy) > x(log(x+1) - logx)
        log(xy) > xlog(x+1/x)
For large x:
        log(x+1/x) ~ log 1 = 0
=>      log(xy) > 0
thus proving the assumption. This estimate of number of primes between two composites thus directly has bearing on di

Matiyasevich-Davis-Robinson-Putnam theorem implies every recursively enumerable set has a diophantine equation. Compl

Definition:
----------
For a set S and subsets A,B of S, if {A,B} is a disjoint set cover of S and if A has a diophantine equation diophanti
```

```
9778  ------------------------------------------------------------------------------------------------------------------------
9779  Theorem: Existence of Complement Diophantine Equation or Complement Function is Undecidable when neither of the compl
9780  ------------------------------------------------------------------------------------------------------------------------
9781  Proof:
9782  ------
9783  MRDP theorem for Hilbert's tenth problem implies every recursively enumerable set is expressible as values of a dioph
9784
9785  Possibility 1 - Generic - Both complementary sets A and B are recursively enumerable:
9786  ---------------------------------------------------------------------------
9787  If both complementary sets A and B are recursively enumerable, they always have a diophantine polynomial - diophantin
9788  There is a known result which states: if set A and B=S-A are both recursively enumerable, then A is recursive. But th
9789
9790  Possibility 2 - Special - One of the complementary sets A or B is recursively enumerable and the other is recursive:
9791  ------------------------------------------------------------------------------------------------------------------------
9792  Both A and B have a diophantine polynomial. If A is recursively enumerable but not recursive, there is a complement E
9793
9794  Possibility 3 - Special - Both complementary sets A and B are recursive:
9795  ------------------------------------------------------------------
9796  Both A and B are recursive and recursively enumerable => Both A and B have diophantine polynomials and both diophanti
9797
9798  Possibility 4 - Special - Both complementary sets are non recursively enumerable:
9799  ---------------------------------------------------------------------------
9800  Obviously both sets A and B are beyond Chomsky hierarchy and there is no algorithm for construction of diophantine(A)
9801
9802  Possibility 5 - Special - One of the complementary sets is recursively enumerable and the other is non recursively en
9803  ------------------------------------------------------------------------------------------------------------------------
9804  One of the sets A is recursively enumerable and thus has a diophantine polynomial. But there exists a complement B=S-
9805
9806  Proof in one line: Any set is a complementary set of some other set and thus any complementary set which is recursive
9807
9808  Construction of a complement function for complementary set:
9809  ------------------------------------------------------------
9810  Construction of a complement function is to find a mapping function f defined as:
9811         f(0) = a1
9812         f(1) = a2
9813         f(2) = a3
9814         ...
9815         f(n) = an
9816  for a1,a2,a3,...,an in Diophantine complementary set, which is equivalent to definition of recursively enumerable tot
9817         f(x) - a = 0
9818  Function f can internally be any mathematical function and can have additional parameters besides x. Finding the enum
9819
9820  An important example for applicability of function complementation is ABC Conjecture which is defined in references b
9821
9822  References:
9823  ----------
9824  472.1 Jones-Sato-Wada-Wiens Theorem and Prime valued Polynomial - https://www.maa.org/sites/default/files/pdf/upload_
9825  472.2 Matijasevic Polynomial - http://primes.utm.edu/glossary/xpage/MatijasevicPoly.html - negative values can be rem
9826  472.3 Hilbert's Tenth Problem and Matiyasevich-Robinson-Davis-Putnam (MRDP) Theorem - https://en.wikipedia.org/wiki/H
9827  472.4 Primes are nonnegative values of a polynomial in 10 variables - [Yu.V.Matijasevic - https://logic.pdmi.ras.ru/~
9828  472.5 Pell's Equation - https://en.wikipedia.org/wiki/Pell%27s_equation - Diophantine equation for set of nonsquare i
9829  472.6 Non-Recursively Enumerable Languages - https://www.seas.upenn.edu/~cit596/notes/dave/relang8.html - powerset of
9830  472.7 Goedel's First Incompleteness Theorem Follows From MRDP theorem - https://en.wikipedia.org/wiki/G%C3%B6del%27s_
9831  472.8 What is Mathematics: Goedel Theorem and Around - https://dspace.lu.lv/dspace/bitstream/handle/7/5306/Podnieks_W
9832         (*) Is set B recursively enumerable? (There are sets which are not recursively enumerable e.g set of subsets o
9833         (*) If set B is recursively enumerable, B has an equivalent diophantine polynomial. But there exists a set B w
9834         (*) In both possibilities, there is a set which does not have a diophantine polynomial - there is a non-recurs
9835  472.9 Goedel Incompleteness and MRDP theorem - https://plato.stanford.edu/entries/goedel-incompleteness/#HilTenProMRD
9836         MRDP Theorem
9837                 There is no general method for deciding whether or not a given Diophantine equation has a solution. ..
9838  472.10 Special Case of Complement Functions and MRDP Theorem - http://www.logicmatters.net/resources/pdfs/MRDP.pdf -
9839  472.11 MRDP Theorem, Jone-Sato-Wada-Wiens Polynomial and Undecidability in Number Theory - [Bjorn Poonen] - http://ww
9840  472.12 Formulas for Primes - https://oeis.org/wiki/Formulas_for_primes#Solutions_to_Diophantine_equations
9841  472.13 Simplest Diophantine Representation - [Panu Raatikainen] - https://pdfs.semanticscholar.org/cd96/ead1a00b73ecc
9842  472.14 Waring Problem and Diophantos/Bachet/Lagrange Four Square Theorem for real quarternions - Topics in Algebra -
9843  472.15 Complement Functions, Diophantine Analysis and ABC Conjecture - https://en.wikipedia.org/wiki/Abc_conjecture -
9844  Let a + b = c be a mutually coprime integer triple f(a,b,c). Quality q(a,b,c) is defined as = log(c)/log(radical(abc)
9845  472.16 Erdos-Straus Diophantine Conjecture - https://en.wikipedia.org/wiki/Erd%C5%91s%E2%80%93Straus_conjecture - For
9846  472.17 ABC Conjecture Proof-designate - Interuniversal Teichmuller Theory - [Shinichi Mochizuki] - http://www.kurims.
9847  472.18 Exponential Diophantines, Fibonacci and Lucas Sequences, Maximum limit on solvability of Diophantine - Reducti
9848
9849  ------------------------------------------------------------------------------------------------------------
9850  473. (FEATURE-DONE) Support Vector Machines Update and Discrete Hyperbolic Factorization Spark Benchmarks
9851  - 14 November 2017
9852  ------------------------------------------------------------------------------------------------------------
9853  (*) Support Vector Machines implementation has been updated to persist the learnt support vectors to
9854   a disk file SupportVectorMachines.txt
9855  (*) This text file is read in classify()
9856  (*) Support Vectors Dictionary is JSON dumped and eval()-ed and not JSON loaded because of serialization
9857    glitch in defaultdict(list)
9858  ----------------------------------------
```

```
(*) Computational Geometric Hyperbolic Factorization Spark implementation has been benchmarked for
 2 more integers of 9 and 10 bits. Some 8 bit numbers were benchmarked earlier.
(*) Spark logs for these benchmarks have been committed and time duration is calculated as 3 way split
 of real/user/system by time shell utility
(*) These benchmark numbers are on dual core single node Spark cluster
(*) C++/python files for tilings have been updated and pixelated tiles storage text files have been rewritten
----------------------------------------
Note on Factorization Spark benchmarks
----------------------------------------
Following are approximate correlations of the observed numbers to the theoretical polylogarithmic time bound - expone

10 bit - real  17m11.931s = 1031.931s (~k*10^x3) ~ (logN)^3.013 (x3=3.013)
9 bit  - real  7m46.247s  = 466.247s  (~k*9^x2) ~ (logN)^2.796 (x2=2.796)
8 bit  - real  3m40.091s  = 220.091s  (~k*8^x1) ~ (logN)^2.589 (x1=2.589)
8 bit  - real  3m39.539s  = 219.539s  (~k*8^x1) ~ (logN)^2.589 (x1=2.589)
8 bit  - real  3m38.795s  = 218.795s  (~k*8^x1) ~ (logN)^2.589 (x1=2.589)
8 bit  - real  3m38.622s  = 218.622s  (~k*8^x1) ~ (logN)^2.589 (x1=2.589)
8 bit  - real  3m38.920s  = 218.920s  (~k*8^x1) ~ (logN)^2.589 (x1=2.589)

----------------------------------------------------------------------------------------------------------------------
474. (THEORY) Computational Geometric Hyperbolic Factorization, Discrete Geometry, Rastering in Graphics/Computationa
Bresenham's Line Algorithm adapted for Hyperbolic tiling, Point Location, Ray shooting - 15,16 November 2017 - relate
----------------------------------------------------------------------------------------------------------------------
Finding factors of integer N by creating pixelated polygon for hyperbolic curve xy=N has great visual intuition. Simi
exist in discrete geometry and computer graphics disciplines. Bresenham's Line drawing algorithm is a classic used st
which approximates a continuous line on a pixelated digital space. This approximation of continuous curves on digital
Rastering. Tile segments of hyperbola in preprocessing step of factorization are found by solving for deltay in:
               xy = N
               (x+1)(y-deltay) = N
               xy - x*deltay + y - deltay = N
               y = deltay * (x+1)
               deltay = y/(x+1) for deltax=1
which is similar to Bresenham Line algorithm for finding next point to plot on raster. Tile segment (N/x, N/(x+1)) al
interval (y, y-(y/x+1)) on y-axis.

Tiling preprocessing phase of factorization embeds a continuous hyperbola in a grid of horizontal and vertical straig
line corresponds to an integer in y-axis and vertical line to an integer in x-axis. Hyperbolic arc traverses the squa
Thus the polygon approximating the continuous hyperbola is the union of all squares(pixels) through which hyperbolic
squares create rectangular faces of the art gallery polygon vertices of which are the locations of the guards. Vertic
through which hyperbolic arc passes through are the factors.

Art Gallery Pixelated polygon approximating a hyperbola is a Planar Simple Line Graph (PSLG) where each side of the p
PSLG. This PSLG has (number_of_factors + 1) rectangular faces which is O(loglogN). Vertices where two adjacent rectan
meet are the factor points of N. Planar point location has to find these factor points. Geometric Ray Shooting Query

References:
----------
474.1 Bresenham Algorithm for Line Rastering - https://en.wikipedia.org/wiki/Bresenham%27s_line_algorithm
474.2 Rasterizing curves - http://members.chello.at/easyfilter/bresenham.pdf
474.3 Efficient Algorithms for Ray Shooting Queries - [Pankaj K.Agarwal] - https://epubs.siam.org/doi/pdf/10.1137/022
474.4 Parallel Planar Point Location - [Richard Cole] - https://ia601408.us.archive.org/33/items/onoptimalparalle00cc
474.5 Parallel Geometric Search - [Albert Chan,Frank Dehne, Andrew Rau-Chaplin] - https://web.cs.dal.ca/~arc/publicat
474.6 Parallel Computational Geometry - https://www.cs.princeton.edu/~chazelle/pubs/ParallelCompGeom.pdf - [A.Aggarwa

----------------------------------------------------------------------------------------------------------------------
475. (FEATURE-DONE) Computational Geometric Factorization - Tiling Update and benchmark numbers for factoring few int
----------------------------------------------------------------------------------------------------------------------
(*) Existing hyperbolic tiling code depends on C++ code in cpp-src/miscellaneous
(*) To remove this dependency, python script for tiling the hyperbolic arc with simple pixelation similar to bresenha
(*) This function is invoked in DiscreteHyperbolicFactorizationUpperbound_Bitonic_Spark.py before bitonic sort
(*) DiscreteHyperbolicFactorizationUpperbound_Bitonic_Spark.py is parametrized and accepts number to factorize as com
      #$SPARK_2.1.0_HOME/spark-submit DiscreteHyperbolicFactorizationUpperbound_Bitonic_Spark.py <number_to_factoriz
(*) Few more time shell builtin benchmark numbers for factorizing 3-bit, 4-bit, 5-bit, 6-bit , 7-bit integers have be

----------------------------------------------------------------------------------------------------------------------
476. (FEATURE-DONE) Computational Geometric Factorization Update - Spark Accumulators, JSON for Tiling etc., - 19 Nov
----------------------------------------------------------------------------------------------------------------------
(*) Hyperbolic tiling code in python has been changed to reflect C++ tiling in cpp-src/miscellaneous
(*) globalcoordinates global variable has been made Spark Accumulator Mutable global state - globalmergedtiles is alr
(*) bitoniclock acquire/release statements have been added for global variables, but commented for benchmarking
(*) New boolean flags for bitonic comparator python style variable swap, for enabling multiple threads for assign hav
(*) merge_sorted_halves() has been invoked as a separate function
(*) C++ tiling has been chosen because accuracy of long double in creating tile pixels is better than similar tiling
(*) Benchmark numbers for factoring 511 has been added to testlogs/ and are similar to previous numbers
(*) .mergedtiles and .coordinates files are loaded/dumped as JSON in python hyperbolic tiling

----------------------------------------------------------------------------------------------------------------------
477. (FEATURE-DONE and THEORY) Computational Geometric Factorization Update - Benchmarks and Tiling - 20 November 201
----------------------------------------------------------------------------------------------------------------------
(*) Some further changes to Python hyperbolic tiling have been made - tile endpoints have been cast from floating poi
for create_tile()
```

(*) C++ tiling in cpp-src/miscellaneous/DiscreteHyperbolicFactorizationUpperbound_Bitonic.cpp has been parametrized a
as commandline argument the integer to factorize.
(*) Shell script cpp-src/miscellaneous/DiscreteHyperbolicFactorizationUpperbound_Bitonic.sh has been changed to pass
factorize to the C++ tiling binary in cpp-src/ and spark-submit for PySpark executable in python-src/ from shell args
(*) This shell script unifies C++ tiling and PySpark factorization
(*) Known issues: Python tiling still differs from C++ tiling significantly. Changing the hardcoded tile arrays stora
to malloc()-ed heap storage causes faulty tiling. There are only optimization issues which do not affect the factoriz
works well and more benchmarks were done e.g for integers 723 and 501. Numbers for these and past numbers are charted
(*) Following profiling numbers along with previous benchmarks for 8,9,10 bits (14 November 2017) capture the trend i
(*) Gradual increase in exponent of number of bits because of static number of parallel RAMs (cores) is not quite ste
3-bit to 10-bit integers as one might expect. Integers of same bit numbers need almost equal duration to factorize wh
could be a function of logN and not N. Htop shows equal loading of both cores of CPU and consumption is almost 100%+1
(*) Cloud computing is not exactly a parallel RAM but multiple cores are PRAMs having concurrent access to a shared m
cloud have local memory processing too. Spark's Global State Variables (Accumulators) which are reflected across all
software simulations of Parallel RAMs - same global state is concurrently accessed by CPUs of spark nodes. Present Py
 does the sorting on accumulators. Binary search is not necessary because Maximum elements at the end of the k-merge
is an update to drafts in 34.1 and 34.2:
        - http://sourceforge.net/projects/acadpdrafts/files/DiscreteHyperbolicPolylogarithmicSieveForIntegerFactoriz
        - http://sourceforge.net/projects/acadpdrafts/files/DiscreteHyperbolicPolylogarithmicSieveForIntegerFactoriz

(*) Following are only representative figures and ideal benchmark requires a Spark cloud preferably on machines of hi
 and comparison with existing General Number Field Sieve implementations (which is quasipolynomial-exponential).
---
723 - 10-bit - real   16m30.346s = 990.346s ~ O(logN^2.9956)
511 - 9-bit - real   7m46.518s = 466.518s ~ O(logN^2.8382)
511 - 9-bit - real   7m35.642s = 455.642s ~ O(logN^2.7854)
501 - 9-bit - real   7m36.879s = 456.642s ~ O(logN^2.7864) [shell script - includes time duration for C++ tiling]
100 - 7-bit - real   2m33.911s = 153.911s ~ O(logN^2.5851)
123 - 7-bit - real   2m52.704s = 172.704s ~ O(logN^2.6482)
63 - 6-bit - real   1m17.460s = 77.460s ~ O(logN^2.4243)
33 - 6-bit - real   0m58.950s = 58.950s ~ O(logN^2.2757)
14 - 4-bit - real   0m19.927s = 19.927s ~ O(logN^2.1609)
12 - 4-bit - real   0m19.651s = 19.651s ~ O(logN^2.1609)
6 - 3-bit - real   0m15.118s = 15.118s ~ O(logN^2.4649)

---
478. (THEORY) Computational Geometric Factorization - Tiling Optimizations - K-Merge Sort is dispensable and Local Ti
- 21 November 2017

Finding factor vertices in pixelated hyperbolic polygon is equivalent to geometric search query: "Is there a right tu
downturn in the polygon?". These turning points coinciding with hyperbolic arc are factor points. Number of rectangle
polygon = O(loglogN). Each tile along y-axis in the polygon (array of pixels) is of length deltay = N/[x(x+1)]. Maxim
in the pixelation can be derived as:
        xy=N
        (x+deltax)(y-deltay)=N
        => xy + y*deltax - x*deltay - deltax*deltay = N
        But deltax = 1,
        => y - (x+1) deltay = 0
        deltay = y/(x+1) and y = N/x
        => N/x(x+1) = deltay > 1 [each tile should be of length atleast 1]
        N > x*x + x
        => x = (sqrt(1 + 4N) - 1) / 2
        For large N, sqrt(1 + 4N) ~ sqrt(4N) = 2*sqrt(N)
        => x ~ sqrt(N) for large N
Maximum number of tiles in the pixelation is O(sqrt(N)). If number of processors is O(sqrt(N)), each tile can be assi
tiling can be done in O(1) parallel time. Number of processors required can be reduced by having O(sqrt(N)/(logN)^c)
tiles can be created in O((logN)^c) total parallel time i.e in each iteration O(sqrt(N)/(logN)^c) tiles can be create

Thus factors can be found just by:
        478.1 Tiling in parallel requiring O((logN)^c) tile and O(sqrt(N)/(logN)^c) processors where number of tiles
        478.2 Local Binary Search per processor on each sorted tile of O(logN) time
and no k-merge sort is necessary.

Previous optimization reduces number of PRAMs by orders of (logN)^c but increases exponent of parallel time O((logN)^

References:

478.1 Line segment turn detection - http://fileadmin.cs.lth.se/cs/Personal/Rolf_Karlsson/lect9.pdf - For two line seg
its sides as input array of line segments, cross product of adjacent pairs of line segment vectors can be computed in
478.2 Sweepline algorithms - http://www.ics.uci.edu/~goodrich/pubs/ggb-sweep-j.pdf
478.3 Rectangle Stabbing - School on Geometric Computing, IIT Delhi (2010) - http://www.cse.iitd.ernet.in/~ssen/geoms

---
479. (FEATURE-DONE and THEORY) Computational Geometric Factorization and Parallel Tile Search Updates
- related to 34 and 478 - 22 November 2017

(*) cpp-src/miscellaneous/DiscreteHyperbolicFactorizationUpperbound_Bitonic.cpp has been revamped and unnecessary cod
(*) New Spark python script python-src/DiscreteHyperbolicFactorizationUpperbound_TileSearch_Optimized.py has been add
(*) How Spark parallelizes a sequential datastructure on cloud nodes is equivalent to binary search on an array of le
(*) Benchmark numbers for two integers 1011 and 1013 with and without sorting have been committed to testlogs/ and Ti
(*) As evidenced from python-src/testlogs/DiscreteHyperbolicFactorizationUpperbound_TileSearch_Optimized.log.22Novemb

```
(*) Benchmarks for Local Tile Search obviate the requirement for k-merge sorting of pixelated hyperbolic tiles, if pa
(*) Following is the trade-off:
        - Bitonic K-merge sorting of tiles : binary search is not necessary but a tremendous performance drag and ove
        - Local tile search : binary search is necessary only per processor but requires relatively high number of PR

---------------------------------------------------------------------------------------------------
480. (FEATURE-DONE) Computational Geometric Factorization Tiling Optimization - Binary Search for Tile Segments in Sp
---------------------------------------------------------------------------------------------------
(*) Binary Search has been implemented in Spark Tile Search Optimization by parallizing the set of intervals of pixel
(*) By this each tile interval can be binary searched in parallel to find the factor point with no
necessity for global k-merge sorting.
(*) Separate pixelation and tile interval file creation C++ source file has been added in cpp-src/miscellaneous
(*) Arrayless tile creation has been chosen and only intervals are written to a file suffixed as .tileintervals
(*) Shell script which compiles and executes the tile interval creation file and spark interval binary search script
(*) Invoking the shell script as:
        cpp-src/miscellaneous/DiscreteHyperbolicFactorizationUpperbound_TileSearch_Optimized.sh <number_to_factorize)
is sufficient to print the factors
(*) Following are benchmark numbers for factorizing few reasonably high bit integers. Removing K-Merge sort has signi
throughput even for a single node spark cluster on dual core (local[2]):

Factorization of 12093 (14-bit): real  0m29.589s (dual core - local[2])

Factorization of 65532 (16-bit): real  0m44.899s (single core - local[1])
Factorization of 65532 (16-bit): real  0m39.621s (dual core - local[2])


Factorization of 102349 (17-bit): real 1m5.490s (single core - local[1])
Factorization of 102349 (17-bit): real 0m58.776s (dual core - local[2])

Factorization of 934323 (20-bit): real 8m20.017s (single core - local[1])
Factorization of 934323 (20-bit): real 7m37.958s (dual core - local[2])

Factorization of 1343231 (21-bit): real 16m34.759s (single core - local[1])
Factorization of 1343231 (21-bit): real 10m49.218s (dual core - local[2])

(*) Above tile binary search numbers beat bitonic k-mergesort of tiles by many orders of magnitude
(*) logs containing factors of above integers have been committed to testlogs/
(*) This optimization thus effectively supersedes mergesort of tiles and is thus a better PySpark implementation of D
---------------------------------------------
(*) Updated AsFer Design Document - benchmark numbers for both single and dual cores for tile search
in computational geometric factorization
(*) Updated python-src/DiscreteHyperbolicFactorizationUpperbound_TileSearch_Optimized.py for dual cores
(local[2])

---------------------------------------------------------------------------------------------------
481. (FEATURE-DONE) Computation Geometric Factorization - Binary Search for Tiles - Benchmarks
- 24 November 2017
---------------------------------------------------------------------------------------------------
Benchmark numbers for factoring 24 bit integer by Tile BinarySearch Optimization script have been committed to testlo

Factorization of 9333123 (24-bit):
---------------------------------
real    112m24.101s (Spark duration 12:21 to 14:02 = 101minutes = 6060seconds)
user    0m0.000s
sys     0m0.004s

Again binary search of tiles in parallel is way better than k-mergesort of tiles. Largest known numbers
widely used are 128-bit. Trend for different bits indicates, Spark cloud of few multicore nodes is
sufficient to factorize 128-bit integers in few minutes. Previous benchmark uses spark-default.conf files
from python-src/InterviewAlgorithm/ (2GB of heapspace)

---------------------------------------------------------------------------------------------------------------------
482. (THEORY) Computational Geometric Factorization - Parallel Local Binary Search for Tiled Hyperbolic Arc - 27 Nove
- Snir's Theorem for Parallel Search - related to 34,478
---------------------------------------------------------------------------------------------------------------------
Previous optimized factorization with no sorting and only local binary search per tile interval can achieve further s

Tiling preprocessing of hyperbolic arc creates O(N) tiles. [This is proportional to length of hyperbolic arc obtainab
calculus = DefiniteIntegral(sqrt(1 + [N^2/x^4]))]

Number of iterations = O((logN)^k)
Number of PRAMs = (N/(logN)^k)

Factorization in parallel has following algorithm:
while(iterations <= O((logN)^k))
{
       *) Assign N/(logN)^k tiles to N/(logN)^k PRAMs in parallel (O(1) parallel time because each interval tile in a
       *) Binary Search tile in each PRAM for factors (O(logN) parallel time which can be reduced to O(logN/logp) by
}

Previous loop totally is of [O(1) + O(logN)]*O((logN)^k) = O((logN)^(k+1)) parallel time.For minimum value of k=1, Fa
```

```
local binary search of tiled hyperbolic arc, can be done in O((logN)^2) parallel time and O(N/logN) PRAM processors w
O(logN*logN/logp) in CREW PRAM.

References:
----------
482.1 Efficient Parallel Algorithms and subclasses of NC - [KruskalRudolphSnir] - https://ac.els-cdn.com/030439759096
482.2 On Parallel Search - Snir's Theorem - https://pdfs.semanticscholar.org/3a58/58e8517f28fa586364daffb34160c437bf7
482.3 Point in Polygon (PIP) problem - https://en.wikipedia.org/wiki/Point_in_polygon - queries if a point is inside,
482.4 Simulation of BSP and CRCW PRAM in MapReduce - Sorting, Searching, and Simulation in the MapReduce Framework -
482.5 Models of Parallel Computation - PRAM shared memory model can be simulated on BSP distributed memory model - ht
482.6 PRAM memory access is unit time - [Guy Blelloch] - http://www.cs.cmu.edu/afs/cs/academic/class/15499-s09/www/sc



-----------------------------------------------------------------------------------------------------------------------
483. (FEATURE-DONE) Support Vector Machines - Mercer Theorem - Kernel Implementation - 29 November 2017
-----------------------------------------------------------------------------------------------------------------------
(*) This commit implements the kernel trick for lifting points in lower dimension to higher dimension by a Feature ma
 decision hyperplane in higher dimension separates the points accurately.
(*) Mercer Theorem creates a kernel function unifies Feature map lifting and Dot product in higher dimension
(*) Feature map phi maps a point in a dimension d to a point in dimension d+k: phi(x) = X. Inner Product (Dot) of two
d+k = phi(x)*phi(y). Mercer Theorem unifies the Feature map and Dot product into a Kernel function defined as series:
   K(x,y) = Sigma(eigenvalue(i)*eigenfunction(x)*eigenfunction(y))
(*) This implementation randomly instantiates a N*N square matrix, finds its Eigenvalues and Eigenvectors, and comput
K(x,y) as per previous identity (neglecting imaginary parts of Eigenvalues and Eigenvectors)
(*) logs for this have been added to testlogs/

-----------------------------------------------------------------------------------------------------------
484. (FEATURE-DONE) Support Vector Machines - Mercer Kernel Update - 30 November 2017
-----------------------------------------------------------------------------------------------------------
(*) Mercer Kernel Function has been changed to return a tuple of feature mapped points and the dot product
(*) Feature mapped points in higher dimension are : [....., square_root(eigenvalue[i])*eigenfunction(x[i]),....]

-----------------------------------------------------------------------------------------------------------
485. (FEATURE-DONE) Compressed Sensing - Image Sketch implementation - 1 December 2017
-----------------------------------------------------------------------------------------------------------
(*) Sketch B of an image bitmap X is computed by multiplying with a random matrix A: B=AX
(*) import ImageToBitMatrix from image_pattern_mining/ for mapping an image to a bitmap matrix
(*) Sketch matrix B contains compressed information of the larger image. Original image can be sensed from this sketc

-----------------------------------------------------------------------------------------------------------------------
486. (FEATURE-DONE) Compressed Sensing Update - Decompression and Error estimation of recovered image bitmap from ske
-----------------------------------------------------------------------------------------------------------------------
(*) Sketch Ax=B of an image bitmap has been persisted to a file CompressedSensing.sketch
(*) A is a random matrix of dimensions (m,n) m<<n and m is scaled by a sketch ratio variable
(*) Original image x is recovered from sketch Ax=B by inverting A and multiplying with sketch:
                Ainverse.Ax = x_recovered
(*) For non-square only approximate pseudo inverse is computable.
(*) For inverting non-square matrix A, Moore-Penrose Pseudoinverse function pinv() from NumPy is invoked.
(*) Error of the recovered image computed by trace (sum of all entries) of the recovered image.
(*) Logs for multiple sketch ratios (row values:100,200,300,400,50) have been committed to testlogs/ which show an ir
as size of the sketch decreases.

-----------------------------------------------------------------------------------------------------------------------
487. (THEORY) Computational Geometric Factorization, Tile Search Optimization and Parallel Interval/Segment Search Tr
and 22 December 2017 - related to 34, 482
-----------------------------------------------------------------------------------------------------------------------
Tile Search optimization for finding factors of an integer described earlier, binary-search set of intervals in paral
This is the classic segment/interval search tree problem in Computational Geometry. Segments/Intervals are 1-dimensic

Unique Prime Factorization of an integer N = p1^k1 * p2^k2 ... * pn*kn
This can be rewritten as:
        2^logN = 2^(k1*log(p1) + k2*log(p2) + ... + kn*log(pn))
=>      logN = (k1*log(p1) + k2*log(p2) + ... + kn*log(pn))
        SmallOmega(N) = number of distinct prime factors of N = O(loglogN) from Hardy-Ramanujan Theorem
        BigOmega(N) = number of prime factors including multiplicity = sum of prime powers = k1 + k2 + k3 + ... + kn
But     k1 + k2 + k3 + ... + kn < (k1*log(p1) + k2*log(p2) + ... + kn*log(pn)) = logN
=>      BigOmega(N) < logN
        BigOmega(N) = O(logN) [This is very high upperbound and not tight estimate]

Segment binary search tree representation of pixelated hyperbolic tile segments is described in references below.

Factorization algorithm for N based on Segment Tree Search is:
        (#) Construct Segment Tree of Pixelated Hyperbolic Segments in Parallel (O(logN) or between O((logN)^2) and O
        (#) Query the Segment Tree for factor points in O(k + logN) where k is the number of segment intervals to be

Thus there are 3 Computational Geometric Factorization algorithms described in this draft and all of them are PRAM al
        (#) K-MergeSort and BinarySearch of Hyperbolic tile segments (mentioned in 34 - requires merge sort of tile s
        (#) Local Binary Search in parallel of hyperbolic tiles without K-MergeSort (mentioned in 481 - recent optimi
        (#) Segment Tree Representation and Binary Search of hyperbolic tile Segments (uses a classic datastructure,

Note: First and Third factorization algorithms are equivalent because both involve sorting of pixelated hyperbolic ti
```

```
10183
10184  An Unsorted Search algorithm has been implemented in NeuronRain (Section 500) to locate a query point on an unsorted
10185
10186  Length of each tile on x-axis can be derived from equating for N:
10187          xy = (x+delta)(y-1)
10188          delta = x/(y-1) = N/[y(y-1)]
10189
10190  Sum of lengths of all pixelated hyperbolic tiles along x-axis is the series:
10191          N/(1*2) + N/(2*3) + N/(3*4) + ...
10192  But:
10193          N/(1*2) + N/(2*3) + N/(3*4) + ... < N/1 + N/2^2 + N/3^2 + ... < N + DefiniteIntegral_2_to_N(dx/x^2) = 1.5N-1
10194  => Sum of lengths of all pixelated hyperbolic tiles is upperbounded by 1.5N-1 or O(N)
10195
10196  Some more optimizations:
10197  ------------------------
10198  [#] Sorting is required only if end points of two adjoined segments are in conflict (only some, not all, elements in
10199
10200  [Subsegment1] Elements of succeeding tile segment become bigger than last element in preceding tile segment if:
10201          xy < (x + delta)(y-1)
10202          xy < xy - x + delta(y-1)
10203          => delta > x/(y-1)
10204
10205  [Subsegment2] Similarly first element in succeeding tile is bigger than last element in Subsegment1 if:
10206          (x+delta)y < (x + N/(y-1)(y-2))(y-1)
10207
10208  Splitting each tile segment of length l into two subsegments of length delta and l-delta and binary searching two set
10209
10210  [#] If the quadrant containing hyperbolic arc is partitioned by parallel ray shooting queries from origin, separated
10211          y(m) = SquareRoot(N/[tan(m*pi/(2*k*loglogN))]) - 1 for m=1,2,3,...,kloglogN
10212  These are only approximate angles. Factors should lie in close proximity of the intersection points of these rays wit
10213
10214  References:
10215  -----------
10216  487.1 Parallel Segment Trees - [AV Gerbessiotis] - https://web.njit.edu/~alexg/pubs/papers/segment.ps.gz
10217  487.2 Distributed Segment Trees - [Guobin Shen, Changxi Zheng, Wei Pu, and Shipeng Li - Microsoft Research] - http://
10218  487.3 Computational Geometry - Algorithms and Applications - [Mark de Berg, Otfried Cheong, Marc van Kreveld, Mark Ov
10219  487.4 Parallel Construction of Binary Search Trees - [MJAtallah,SRKosaraju,LLLarmore,GLMiller,SHTeng] - https://www.c
10220  487.5 Segment Trees - definition and diagrams - [Computational Geometry Course Notes - Antoine Vigneron - King Abdull
10221  487.6 Hardy-Ramanujan Theorem - https://en.wikipedia.org/wiki/Hardy%E2%80%93Ramanujan_theorem
10222  487.7 Parallel Construction of Segment Trees - https://www.cs.dartmouth.edu/~trdata/reports/TR92-184.pdf - [Peter Su,
10223  487.8 Parallel Computational Geometry - Parallel Construction of Segment Trees is O(logN) time - Section 5 - Pages 30
10224  487.9 Small omega - Number distinct prime factors of an integer - https://oeis.org/A001221 - is O(loglogN) from Hardy
10225  487.10 Big omega - Number prime factors of an integer with multiplicity - https://oeis.org/A001222 - is sum of prime
10226  487.11 Parallel Construction of Segment Trees applied in another setting - [Helmut Alt, Ludmila Scharf] - computing d
10227  487.12 Function plot of Number of PRAM processors - N/(logN)^k - For k=1, N/logN has been plotted in https://sourcefo
10228  487.13 Interval Hash Trees - [T. F. Syeda-Mahmood, P. Raghavan, N. Megiddo] - http://www.almaden.ibm.com/cs/people/st
10229  487.14 Efficient Parallel Algorithms for Geometric Clustering and Partitioning Problems (1994) - [Amitava Datta] - ht
10230  487.15 Parallel computational geometry of rectangles - [Chandran-Kim-Mount] - https://link.springer.com/article/10.16
10231
10232  ------------------------------------------------------------------------------------------------------------------------
10233  488. (THEORY) Thermodynamics Gas Diffusion Entropy Model of Information Diffusion in Social Networks and Text Graphs
10234  - related to 384
10235  ------------------------------------------------------------------------------------------------------------------------
10236  Second Law of Thermodynamics implies entropy of a closed system always increases. Gas diffusion in a closed chamber d
10237  at random with in the space of the chamber. Gas chamber is a 3-dimensional grid of volume N. If in initial state, all
10238
10239  References:
10240  ----------
10241  488.1 Solomon Asch Conformity Experiments and Herd behaviour - Information Diffusion in Social Media - Section 7.1 -
10242  488.2 Emperor's New Mind - [Roger Penrose] - Inexorable increase in entropy - Page 405
10243
10244  ------------------------------------------------------------------------------------------------------------------------
10245  489. (FEATURE-DONE) NeuronRain AsFer-KingCobra MAC Electronic Money - Proof-of-Work and Universally Unique ID Hash im
10246  14 December 2017
10247  ------------------------------------------------------------------------------------------------------------------------
10248  (#) Fictitious Message-As-Currency (MAC) in AsFer-KingCobra has been named "Neuro".
10249  (#) This commit implements a non-trivial proof-of-work computation and finds a universally unique hash id for each Ne
10250   which has 2 leading "ff"s - Proof of Work is analogous to reCAPTCHA in websites for filtering out robots
10251  (#) Unique id is created from Boost UUID random generator and checked in a loop for 2 leading "ff" in stringified hex
10252  (#) Protocol Buffer version has been upgraded to 3.5/15 and currency.proto has been recompiled with protoc
10253  (#) asfercloudmoveclient.cpp has been updated to make a choice between std::move() and std::forward() move semantics:
10254       - std::move() just does =operator overload and moves Neuro currency over network
10255       - std::forward() + std::move() first deferences rvalue of && for currency uuid and then does =operator overload
10256  (#) Limitation: Presently there is no documented way to create an rvalue for non-primitive datatypes. For example stc
10257  as literal string "xxxxxx".
10258
10259  ------------------------------------------------------------------------------------------------------------------------
10260  490. (FEATURE-DONE) AsFer-KingCobra Neuro Electronic Currency - Rvalue NonPrimitive Perfect Forward - 18 December 201
10261  ------------------------------------------------------------------------------------------------------------------------
10262  (#) New constructor for cloudmove which takes const char* uuid has been defined. This enables
10263  assigning a string literal rvalue to cloudmove<currency::Currency> objects.
```

10264 (#) New move operator= which takes const char* uuid has been defined. This internally instantiates
10265 a currency::Currency object
10266 (#) New move operator= which takes cloudmove<T>& lvalue has been defined.
10267 (#) New value and a clause for move semantics, "nonprimitiveforward" has been defined. This clause
10268 does std::forward() of an rvalue cloudmove<currency::Currency> and then invokes std::move() of the currency over netw
10269 (#) This differs from move semantics "std::forward" which is specific to std::string uuid only, and thus solves the g
10270 (#) client and server logs for this network move have been committed to testlogs.
10271
10272 ----------
10273 491. (THEORY) Generalization of Complementation Undecidability to Rationals(Q) and Reals(R) by H10(Hilbert Tenth Prob
10274 Connection between ZF with Axiom of Choice (ZFC) and Complementation - related to 472 - 28 December 2017, 29 March 20
10275 ----------
10276 Thus far Function Complementation described in drafts of this document are oriented towards set of natural numbers or
10277
10278 Axiom of Choice: For set of sets S={s1,s2,s3,...} there exists a choice function C(si)=xi which chooses an element xi
10279
10280 Following is a special case example of Axiom of Choice in Boolean Social Choice functions:
10281 Depth-2 boolean function/circuit B leaves of which are sets s1,s2,s3,.... and nodes at level one are C(s1)=x1,C(s2)=x
10282
10283 Disjoint Set Cover is also known as Exact Cover defined as subcollection S' of collection of subsets S of a universal
10284 X is contained in exactly one subset in S'. S' is a disjoint collection and the exact set cover of X. Exact Cover pr
10285 solved by DLX Dancing Links algorithm. Complementary or Ramsey k-colored sets are created by Exact Set Cover.
10286
10287 References:
10288 ----------
10289 491.1 Diophantines for Reals - http://wwwmayr.in.tum.de/konferenzen/Jass07/courses/1/Sadovnikov/Sadovnikov_Paper.pdf
10290 491.2 Uncountable sets and non-recursively enumerable languages - http://www.cs.colostate.edu/~massey/Teaching/cs301/
10291 491.3 Deciding the Undecidable - https://books.google.co.in/books?id=1rjnCwAAQBAJ&pg=PA10&lpg=PA10&dq=real+solutions+
10292 491.4 Introduction to Automata Theory, Languages and Computation - [John E.Hopcroft,Rajeev Motwani,Jeffrey D.Ullman]
10293 491.5 Real Roots of polynomials - https://en.wikipedia.org/wiki/Root-finding_algorithm
10294 491.6 Constructing Diophantine Representation of a Listable Set, Register Machines preferred over Turing Machines - h
10295 491.7 Diophantine Representation, Non-Deterministic Diophantine Machine (NDDM), Complexity class NP, Single Fold Diop
10296 491.8 DPR theorem and finite-fold diophantine representations - [Yuri Matiyasevich] - ftp://ftp.pdmi.ras.ru/pub/publi
10297 491.9 Factoring Semi-primes (numbers of the form N=pq for prime p,q) by Diophantine Equations - http://www2.mae.ufl.e
10298 491.10 Dancing Links X Algorithm For Exact Cover - Pentominoes Example - 4-way doubly linked list - [Donald E. Knuth]
10299 491.11 Dancing Links - [Hitotumatu, Hirosi; Noshita, Kohei] (1979). "A Technique for Implementing Backtrack Algorithm
10300
10301 ----------
10302 492. (THEORY and FEATURE-DONE) Complement Function Map Construction - Diophantine Representation - Lagrange's Four Sq
10303 (Draft updates to: https://arxiv.org/abs/1106.4102) - related to 472 - 4 January 2018
10304 ----------
10305 (#) This commit implements diophantine representation of an enumerable recursive set by Sum of Four squares solver
10306 in SymPy.
10307 (#) New function that enumerates each element x of the complement set and applies it as a parameter to Sum of Squares
10308 to obtain the quadruple (a,b,c,d) such that a^2 + b^2 + c^2 + d^2 = x
10309 (#) This is mostly partial recursive function and not necessarily a total recursive function(defined for all possible
10310 (#) logs for this have been committed to testlogs/
10311 (#) SymPy has other diophantine solvers too which require more than one parameters.
10312 (#) SymPy does not have exponential diophantine support yet which if available could construct an exponential diophar
10313 recursively enumerable set ('almost' because of MRDP theorem - there are diophantine equations for recursively enumer
10314
10315 Any partial function f:X - Y which maps a subset X' of X to Y, can be converted to a total function by mapping all el
10316
10317 References:
10318 ----------
10319 492.1 Reduction of arbitrary diophantine equation to one in 13 unknowns - [Matijasevic-Robinson] - http://matwbn.icm.
10320 492.2 Distinction between Enumerable and Effectively Enumerable - http://faculty.washington.edu/keyt/Effenumerability
10321 (*) first 13-tuples are found by a diophantine solver and
10322 (*) these tuples are listed as t(0),t(1),t(2),....
10323 492.3 Function Spaces - Extending partial function to a total function - https://en.wikipedia.org/wiki/Partial_functi
10324
10325 ----------
10326 493. (FEATURE-DONE) Computation Geometric Factorization - Binary Search for Tiles - Optimization - Benchmarks - Revis
10327 - related to 481
10328 ----------
10329 (#) Factorization benchmarks for Tile binary search optimization have been redone on dual core (local[2]) after remov
10330 (#) Following are the durations for factoring same 24 bit integer factorized previously after removing print statemer
10331 --
10332 Factorization of 9333123 (24-bit) - without print statements
10333 --
10334 real 35m56.196s (Spark Duration = 1839.146seconds)
10335 user 3m2.836s
10336 sys 4m56.668s
10337 --
10338 Factorization of 9333123 (24-bit) - with print statements earlier
10339 --
10340 real 112m24.101s (Spark duration 12:21 to 14:02 = 101minutes = 6060seconds)
10341 user 0m0.000s
10342 sys 0m0.004s
10343 ==
10344 Factors of 9333123 are (excerpt from logs):

```
==================================================
...
Factor is =  219
Factor is =  1387
Factor is =  2243
Factor is =  4161
Factor is =  6729
Factor is =  6729
Factor is =  42617
Factor is =  127851
Factor is =  163739
Factor is =  491217
Factor is =  3111041
Factor is =  9333123
==================================================
(#) Compressed Spark log for this factorization has been committed to python-src/testlogs/
(#) General Number Field Sieve algorithm is sequential, exponential in number of bits and requires O(2^(c(logN)^0.33*

----------------------------------------------------------------------------------------------------------------------
494. (THEORY and FEATURE-DONE) Complement Diophantine Map - Converts a Partial Function to Total Function - 6 January
----------------------------------------------------------------------------------------------------------------------
(#) Sum of Four Squares diophantine solutions are quadruples (a,b,c,d) which form a subset of all possible tuples and
complement function unknows-to-parameter map is partial.
(#) This is remedied by extending the parameter codomain by adding -1 as additional possible parameter value and mapp
(#) logs for this have been committed to testlogs/

----------------------------------------------------------------------------------------------------------------------
495. (THEORY) Factoring as a service - General Number Field Sieve on Amazon EC2 cloud - RSA 512 bit - Relevance to Co
Geometric NC-PRAM-Multicore Factorization Theoretical O((logN)^2) bound - related to 482 - 9 January 2018
----------------------------------------------------------------------------------------------------------------------
Number field sieve has been implemented as Amazon EC2 cloud service. Benchmarks for RSA 512 bit achieve factorization
4 hours applying CADO-NFS and MSieve NFS implementations optimized for Elastic Cloud. Instance type used is c4.8xlarg

References:
----------
495.1 Factoring as a service - slides - http://crypto.2013.rump.cr.yp.to/981774ce07e51813fd4466612a78601b.pdf
495.2 Factoring as a service - https://github.com/eniac/faas - Parallelizing Number Field Seive - Polynomial selectio

----------------------------------------------------------------------------------------------------------------------
496. (THEORY and FEATURE-DONE) Computational Geometric Factorization Update - Local Tile Computation and necessity fo
10 January 2018 - 30-bit integer dual core single node Spark Cluster benchmark - related to 495
----------------------------------------------------------------------------------------------------------------------
As mentioned in previous section, reading tiles from storage in SparkContext.parallelize() is a serious bottleneck. I
---------------------------
Factorization of 921234437:
---------------------------
18/01/10 17:56:56 INFO Utils: /home/shrinivaasanka/Krishna_iResearch_OpenSource/GitHub/asfer-github-code/python-src/D
Factor is =  1
Factor is =  17
Factor is =  19
Factor is =  59
Factor is =  323
Factor is =  1003
Factor is =  1121
Factor is =  19057
Factor is =  48341
Factor is =  821797
Factor is =  918479
Factor is =  2852119
Factor is =  15614143
Factor is =  48486023
Factor is =  54190261
18/01/10 18:08:51 INFO PythonRunner: Times: total = 714516, boot = 649, init = 23, finish = 713844

Spark Duration: 18:08:51 - 17:56:56 = 535seconds

Spark-Python has RPC and serialization latencies and following numbers could be better if implemented in a different

----------------------------------------------------------------------------------------------------------------------
497. (THEORY and FEATURE-DONE) Computational Geometric Factorization - Non-Persistent Tile Segments - 30 bit Single N
----------------------------------------------------------------------------------------------------------------------
(#) Benchmarks for factoring another 30-bit integer 999994437 on single node dual core Spark cluster have been commit
(#) Similar to previous 30 bit integer, time utility prints 40 minutes 56 seconds while the actual Spark duration is
18/01/11 13:34:34 INFO Utils: /home/shrinivaasanka/Krishna_iResearch_OpenSource/GitHub/asfer-github-code/python-src/D
==================================================
Factor is = 1
==================================================
==================================================
Factor is = 3
==================================================
==================================================
```

```
Factor is = 9
================================================
================================================
Factor is = 13
================================================
================================================
Factor is = 23
================================================
================================================
Factor is = 27
================================================
================================================
Factor is = 39
================================================
================================================
Factor is = 69
================================================
================================================
Factor is = 97
================================================
================================================
Factor is = 117
================================================
================================================
Factor is = 207
================================================
================================================
Factor is = 291
================================================
================================================
Factor is = 299
================================================
================================================
Factor is = 351
================================================
================================================
Factor is = 621
================================================
================================================
Factor is = 873
================================================
================================================
Factor is = 897
================================================
================================================
Factor is = 1261
================================================
================================================
Factor is = 1277
================================================
================================================
Factor is = 2231
================================================
================================================
Factor is = 2619
================================================
================================================
Factor is = 2691
================================================
================================================
Factor is = 3783
================================================
================================================
Factor is = 3831
================================================
================================================
Factor is = 6693
================================================
================================================
Factor is = 8073
================================================
================================================
Factor is = 11349
================================================
================================================
Factor is = 11493
================================================
================================================
Factor is = 16601
================================================
================================================
```

```
Factor is = 20079
================================================
================================================
Factor is = 29003
================================================
================================================
Factor is = 29371
================================================
================================================
Factor is = 34047
================================================
================================================
Factor is = 34479
================================================
================================================
Factor is = 49803
================================================
================================================
Factor is = 60237
================================================
================================================
Factor is = 87009
================================================
================================================
Factor is = 88113
================================================
================================================
Factor is = 123869
================================================
================================================
Factor is = 149409
================================================
================================================
Factor is = 261027
================================================
================================================
Factor is = 264339
================================================
================================================
Factor is = 371607
================================================
================================================
Factor is = 381823
================================================
================================================
Factor is = 448227
================================================
================================================
Factor is = 783081
================================================
================================================
Factor is = 793017
================================================
================================================
Factor is = 1114821
================================================
================================================
Factor is = 1145469
================================================
================================================
Factor is = 1610297
================================================
================================================
Factor is = 2848987
================================================
================================================
Factor is = 3344463
================================================
================================================
Factor is = 3436407
================================================
================================================
Factor is = 4830891
================================================
================================================
Factor is = 8546961
================================================
================================================
Factor is = 10309221
================================================
================================================
```

```
Factor is = 14492673
================================================
================================================
Factor is = 25640883
================================================
================================================
Factor is = 37036831
================================================
================================================
Factor is = 43478019
================================================
================================================
Factor is = 76922649
================================================
================================================
Factor is = 111110493
================================================
18/01/11 13:48:32 INFO PythonRunner: Times: total = 837634, boot = 641, init = 37, finish = 836956
18/01/11 13:48:32 INFO Executor: Finished task 0.0 in stage 0.0 (TID 0). 1529 bytes result sent to driver
18/01/11 13:48:32 INFO TaskSetManager: Starting task 1.0 in stage 0.0 (TID 1, localhost, executor driver, partition 1
18/01/11 13:48:32 INFO Executor: Running task 1.0 in stage 0.0 (TID 1)
================================================
Factor is = 333331479
================================================
18/01/11 13:48:32 INFO TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 838426 ms on localhost (executor dri

which is almost 14 minutes ( 13:48:32 - 13:34:34 ). First non-trivial factor was printed within 10 seconds. Time util

-----------------------------------------------------------------------------------------------------------------------
498. (THEORY and FEATURE-DONE) Computational Geometric Factorization - Non-Persistent Tile Segments - 31 bit Single N
-----------------------------------------------------------------------------------------------------------------------
31-bit integer 2147483647 has been factorized by searching non-persisted pixelated hyperbolic tile segments. Only tri

-------------------------------------------------------------------------
Factorization/Primality of 2147483647
-------------------------------------------------------------------------
18/01/12 14:33:12 INFO Utils: /home/shrinivaasanka/Krishna_iResearch_OpenSource/GitHub/asfer-github-code/python-src/D
================================================
Factor is =  1
================================================
18/01/12 15:02:57 INFO PythonRunner: Times: total = 1784724, boot = 643, init = 23, finish = 1784058
-------------------------------------------------------------------------
Spark Duration 15:02:57 - 14:33:12 = 29 minutes 45 seconds

Spark Logs have been committed to python-src/testlogs/

References:
----------
498.1 Multiplicative Partition Function - ON THE OPPENHEIM���S ���FACTORISATIO NUMERORUM��� FUNCTION - [FLORIAN LUCA,
498.2 Multiplicative Partition Function - https://oeis.org/A001055
498.3 Bound for Multiplicative Partition Function - [Canfield-Erdos-Pomerance] - https://www.math.dartmouth.edu/~carl
498.4 Eigth Mersenne Prime - 2^31 - 1 = 2147483647 - https://en.wikipedia.org/wiki/2,147,483,647 - this was largest k
498.5 M31 - http://primes.utm.edu/curios/page.php/2147483647.html - The first prime that cannot be tested on 32-bit p
498.6 Cryptography in NC0 - [Benny Applebaum, Yuval Ishai, Eyal Kushilevitz] - http://www.cs.technion.ac.il/~abenny/p
498.7 Binary Quadratic Diophantine Equations (BQDE) and BQDE for Factorization - [JC Lagarias] - https://arxiv.org/pd
498.8 Polynomial Time Quantum Algorithm for Solving Pell's Equation - [Hallgren] - http://www.cse.psu.edu/~sjh26/pell
        (x+1)/y = a
        (x-1)/y = b
        2/y = a-b for factors a,b of N.
        y = 2/a-b
        => (x+1) = 2a/(a-b)
        => x = 2a/(a-b) - 1
Historicity of this problem (Brahmagupta's Chakravala) and a polylog approximation based on Newton-Raphson square roo

-----------------------------------------------------------------------------------------------------------------------
499. (FEATURE-DONE) Secure Neuro Currency Cloud Perfect Forward Move - OpenSSL client and server - 19 January 2018
-----------------------------------------------------------------------------------------------------------------------
Neuro Currency Cloud Perfect Forward Move socket code has been openSSL enabled:
- Makefile updated include -DOPENSSL #ifdef option for compiling SSL client-server headers
- license headers updated
- new header asfercloudmove_openssl.h has been added to repository for invoking openSSL client and server functions i
and opensslserver.h
- An example fictitious X.509 cert.pem and key.pem have been created by openssl utility for certificate verification
- new headers opensslclient.h and opensslserver.h have been added to repository which define openSSL client and serve
 are reference examples in www.openssl.org Wiki changed for cloud move)
- logs for SSL cloud_move client and server have been committed to cloud_move/testlogs/

-----------------------------------------------------------------------------------------------------------------------
500. (FEATURE-DONE) Searching Unsorted List of Numbers - Algorithm in GRAFIT Open Learning Implemented - 21 January 2
-----------------------------------------------------------------------------------------------------------------------
An algorithm to search list of numbers better than bruteforce search mentioned in GRAFIT Course Notes:
```

https://github.com/shrinivaasanka/Grafit/blob/master/course_material/ComputerScienceMiscellaneous/ComputerScienceMisc
has been implemented in NeuronRain AsFer.

This lifts one dimensional numbers to multidimensional tuples and creates hashtables for each dimension of tuple. Loc
lookup of each digit in query in these hash tables in succession. If all hash lookups succeed queried number exists i

This implementation pads each number with "#" so that it is right justified and maximum number of digits is stipulate

Searching Unsorted lists is the most fundamental open research problem and this implementation uses Locality sensitiv
. Python dictionaries are hash maps and not hash tables which support collisions. Each digit in query is lookedup for
 there is a match digitmatch is set to True. Logical AND of all digit matches is returned as True/False.

---
501. (THEORY) Answer-Questioning (Reversal of Interview) and Recursive Gloss Overlap Dense Subgraph Classification -
- related to 413, 421

Recursive Gloss Overlap Definition Graph of a text previously has been demonstrated to classify documents in unsuperv
dense subgraphs: centrality, k-cores, pageranks etc., Dense subgraphs of text graph extract the essence of an article
which may or may not be present in the document. Keywords not present in the document can be classes of a document be
deep learning of definitions. From the keywords, a question or set of permutations of questions can be constructed wh
Relevance of Question depends on the relatedness e.g Tensor Neuron of all possible pairs of keyword class vertices.

Example:

If a text graph of an academic research text has been classified in following word vertices of high core numbers
              [Prime, Factorization, Theorem, Composites, Diophantine, Polynomials,...]
in decreasing value of centrality/k-core/pagerank and pairwise Tensor Neuron potentials are ranked as below:
              Prime-Factorization = 0.34234
              Theorem-Composites=0.33333
              Diophantine-polynomials=0.221212
              ... and so on
then question(s) can be constructed based on previous ranking of Tensor relatedness as:
              Does this article discuss relation between [Primes] and [Factorization]?
              ....

Reference:

501.1 CALO, SIRI, PAL - https://pal.sri.com/architecture/ - Cognitive Assistants, Question Answering

---
502. (FEATURE-DONE) Scheduler Analytics - Interprocess Distance Computation by DictDiffer - 24 January 2018
---
1. Representing OS Process Information as Feature Vectors has been mentioned as Software Scheduler Analytics usecase
http://neuronrain-documentation.readthedocs.io/en/latest/
2. This commit implements a new python function which reads process info as a dictionary from psutils. Psutils has an
process metrics or to read the dictionary in its entirety.
3. Psutils per-process dictionary has all the basic details pertaining to instantaneous resource consumptions of a pr
4. These per-process features are collated in an array and are json.dump()-ed as strings in asfer.enterprise.encstr.s
which can be used as input by clustering/classification NeuronRain-AsFer C++ implementations.
5. logs for this have been committed to software_analytics/testlogs/DeepLearning_SchedulerAnalytics.log.24January2018
6. Distance between two process feature dictionaries/vectors are printed by DictDiffer which has been imported and le
 the distance function between any two processes.

---
503. (FEATURE-DONE) Ephemeris Search - Maitreya Text Client version 8.0 - 25 January 2018
(Only in NeuronRain Research - SourceForge)
---
1. python-src/MaitreyaEncHoro_RuleSearch.py has been updated to invoke Maitreya Text Client 8.0 and an option to choo
Maitreya text client 7.0 and 8.0.
2. geonames import has been made optional - only if useGeoNames is True
3. Example rule search logs by maitreya 8.0 text client - chartsummary.rulesearch and python-src/testlogs/MaitreyaEnc

---
504. (FEATURE-DONE) Streaming Algorithms - Encoded Strings Data Source - 26 January 2018
(For Astronomical Encoded Strings)
---
1. python-src/Streaming_AbstractGenerator.py iterator has been updated for new data storage "AsFer_Encoded_Strings" a
for streaming encoded strings created by NeuronRain AsFer
2. Streaming algorithm implementations have been updated to stream data from this new datasource and logs have been c

---
505. (THEORY) Computational Geometric Factorization - Feasibility of Sequential Optimization - related to 487 - 6 Feb
---
Doing away with Parallel RAMs in Computational Geometric Factorization by breaking the hyperbolic arc bow into two se
and searching their concatenations which are implicitly sorted ascending, has problems because concatenation of multi
in O(logN) time by using Rope Strings which are tree representations of strings (logarithmic in length of strings and
strings). But concatenation of multiple strings/tiles logarithmic time in number of strings is still open.

Following sequential optimization tries to find feasibility of doing Sequential Factorization without PRAMs and conca

number_to_factorize=N

```
factor_candidate=N/2

---------------------------
Loop for subsegment1 tree:
---------------------------
while(factor is not found)
{
       1. Find the tile segment/interval containing factor candidate.
       [2. Split the candidate tile segment to two subsegments - subsegment1 contains points which are less than prev
       3. Binary Search each node in subsegment1 tree for factor points (p,q) (N=pq) - There are 3 possibilities:
       3.1 All points in this subsegment1 tree node are less than N => Binary Search has to be done on right subsegme
       3.2 All points in this subsegment1 tree node are greater than N => Search has to be done on left subsegment1 s
       3.3 N is present in this subsegment1 tree node and factor point N=pq is found
}

---------------------------
Loop for subsegment2 tree:
---------------------------
while(factor is not found)
{
       4. Find the tile segment/interval containing factor candidate.
       [5. Split the candidate tile segment to two subsegments - subsegment1 contains points which are less than prev
       6. Binary Search each node in subsegment2 tree for factor points (p,q) (N=pq) - There are 3 possibilities:
       6.1 All points in this subsegment2 tree node are less than N => Binary Search has to be done on right subsegme
       6.2 All points in this subsegment2 tree node are greater than N => Search has to be done on left subsegment2 s
       6.3 N is present in this subsegment2 tree node and factor point N=pq is found
}

7.Above algorithm is Depth-2 Two Level Binary Search:
7.1) First binary search finds the subsegment node in subsegment trees in O(logN)
7.2) Second binary search searches within subsegment node in O(logN)
and thus requires O(logN*logN) sequential time.

8. Sets of subsegment1(s) and subsegment2(s) constitute 2 binary search trees of segments, but these two search trees

9. Finding tile segment containing a factor candidate point is non-trivial planar point location problem in general.
Sum of lengths of k consecutive hyperbolic pixelated tiles = N/(1*2) + N/(2*3) + N/(3*4) + ... + N/(k*(k+1))
       = N(1/1 - 1/2 + 1/2 - 1/3 + 1/3 - 1/4 + ... + 1/k - 1/(k+1))
       = N(1 - 1/(k+1))
       = Nk/(k+1)
Tile containing point x:
       Nk/(k+1) < x < N(k+1)/(k+2)
       k < x/(N-x)
integer round-off of k is the index of tile interval containing x.

10. Previous sequential optimization of O((logN)^2) though removes PRAMs, proves only that Factorization is in P. Com

11. For each segment k in loops previously, 2 subsegments of it have following interval endpoints - (xleft,yleft,xrig
       subsegment1:  (N/(k+2),(k+1),N/(k+2)+delta,(k+1))
       subsegment2:  (N/(k+2)+delta,(k+1),N/(k+1),(k+1))
and delta=N/((k+1)(k+2))

-------------------------------------------------------------------------------------------------------------------
506. (THEORY) Tournament Graph Election, Coloring, Intrinsic Merit, Partitions and Bell Number - 8 February 2018
-------------------------------------------------------------------------------------------------------------------
Tournament Graph of n vertices has n(n-1)/2 edges (complete). Each vertex of the tournament graph is uniquely colored
Each edge (v1,v2) is the contest between vertices v1 and v2. Each edge of the tournament graph is progressively color
all possible partitions of a set is the Bell number. Vertex corresponding to Color of the biggest part in this edge s

-------------------------------------------------------------------------------------------------------------------
507. (FEATURE-DONE) Streaming Analytics Abstract Generator Update - Socket Streaming Datasource Added - 8 February 20
-------------------------------------------------------------------------------------------------------------------
1.Streaming Abstract Generator facade/generator pattern implementation has been updated for a new "Socket Streaming"
2.Constructor Datasource arg is the remote host and port is hardcoded to 64001 (one more than kernel_analytics driver
3.Python socket has been imported and socket client code has been added to __iter__()
4.As example, Hyper LogLog Counter Streaming implementation has been updated to read streaming data from remote strea
5.Logs for this have been committed to testlogs/
6.Example webserver commandline:
        nc -l 64001
        ><data>

-------------------------------------------------------------------------------------------------------------------
508. (FEATURE-DONE) Scheduler Analytics - Socket Streaming Server - Decorator pattern implementation - 9 February 201
-------------------------------------------------------------------------------------------------------------------
1. New Socket Streaming Server for Scheduler Analytics has been implemented in python-src/software_analytics/Schedule
2. This invokes get_stream_data() function in python-src/software_analytics/DeepLearning_SchedulerAnalytics.py
3. get_stream_data() function is decorated by a Decorator class implemented in a new file python-src/webserver_rest_u
4. SocketWebServerDecorator implemented in python-src/webserver_rest_ui/NeuronRain_Generic_WebServer.py is generic an
 __init__() and __call__() functions. __call__() invokes the decoratee function get_stream_data() specific to Schedul
5. SocketWebServerDecorator can decorate any other streaming datasource, not just scheduler analytics - decoratee fun
6. Logs for Socket Streaming Server have been committed to python-src/software_analytics/testlogs/SchedulerAnalytics_
```

```
7. Global configs for socket streaming server host and port (64001) have been set in a new config file python-src/sof
8. Known issue: There seems to be a random iterator disconnect because of psutil process_iter() process statistics st

------------------------------------------------------------------------------------------------------------------------
509. (FEATURE-DONE) Scheduler Analytics Socket Streaming Decorator - Psutil iterator frequent disconnects resolution
------------------------------------------------------------------------------------------------------------------------
1. SocketWebServerDecorator frequently and periodically throws "NoneType object not callable" exception.
2. This exception happens after every 215 or 217 processes (not sure what this magic number is)
3. Because of this try/except error handling has been added throughout decorator code
4. Socket listen queue size has been increased to 100
5. datasourcefunc() is re-called once an exception is thrown in a loop - a palliative cure
6. After this 215 barrier is breached. Logs for 558 processes streamed by Socket Server Decorator and received by Str
Generator client have been committed to python-src/software_analytics/testlogs/SchedulerAnalytics_WebServer.log.11Feb
python-src/testlogs/Streaming_HyperLogLogCounter.log.11February2018

------------------------------------------------------------------------------------------------------------------------
510. (FEATURE-DONE) Approximate 3SAT Solver Randomized Rounding Update - NumPy random choice() replacing permutation(
------------------------------------------------------------------------------------------------------------------------
1. CNFSATSolver.py - function creating random 3SAT instances has been updated to invoke NumPy random choice() instead
permutation() without replacement - equivalent to nP3 per clause for n variables.
2. Logs for 16 variables - 16 clauses and 18 variables - 18 clauses have been committed to testlogs/CNFSATSolver.16va
and testlogs/CNFSATSolver.18variables18clauses.log.27February2018
3. Numpy random choice() is based on Uniform distribution.
4. It is remarkable to note that observed probability average and Random Matrix probability 1/sqrt(m*n)
 are strikingly equal - a confirmation of the least squares randomized rounding and relaxation SAT solver's accuracy
5. Logs for 16*16 and 18*18 have few hundreds and thousand plus random 3SAT instances which are solved 100%

------------------------------------------------------------------------------------------------------------
511. (FEATURE-DONE) ConceptNet5 Update - REST endpoints changed - 27 February 2018
------------------------------------------------------------------------------------------------------------
1. ConceptNet5 rest_client.py has been imported which already wraps REST methods and endpoints
2. REST endpoints for search,lookup and query have been updated for ConceptNet 5.5 which were 5.4 earlier

------------------------------------------------------------------------------------------------------------------------
512. (FEATURE-DONE and THEORY) Approximate SAT Solver and ConceptNet REST client updates - 21 variables,21 clauses LS
------------------------------------------------------------------------------------------------------------------------
1.Least Square SAT solver has been updated to print more readable debug messages and highlight MAXSAT approximation r
 average for SATs solved thus far
2.Number of variables and clauses is set to 21,21 and MAXSAT ratio is 98% for 34 random 3SATs
3.Logs for 21,21 has been committed to testlogs/CNFSATSolver.21variables21clauses.log.28February2018 which shows the
probabilities and random matrix 1/sqrt(m*n) literal probability agreeing well.
4.This is because of uniform unbiased (mostly) choice() when number of clauses = number of variables causing 1/sqrt(m
5.For unequal clauses and variables numbers, an epsilon biased PRG implementation in linux kernel is necessary (randc
 rewritten)
---------------------
6.ConceptNet5 client has been updated to include a related() function wrapper which invokes /related REST endpoint fo
7.ConceptNet5 finds related concepts by word embeddings implementation of word2vec which maps each word to a vector a
are clustered together in this vector space. More precisely, word distances are represented as a word-word 2 dimensio
and sum of path edge weights between word pair is the distance.
8.For finding distance between two concepts a common ancestor algorithm has been implemented in new conceptnet_distan
9.Common ancestors are printed in end and distance between the concepts is also printed
10.Some example conceptnet_distance() invocations have been captured in ConceptNet/testlogs/ConceptNet5Client.log.28F

References:
-----------
512.1 ConceptNet 5.5: An Open Multilingual Graph of General Knowledge - [Robert Speer,Joshua Chin,Catherine Havasi-Lu
512.2 ConceptNet blog - https://blog.conceptnet.io/
512.3 Microsoft Concept Graph - Probase - https://concept.research.microsoft.com/Home/Introduction - similar to Conce

------------------------------------------------------------------------------------------------------------------------
513. (THEORY) ConceptNet, Graph Tensor Neuron Network Intrinsic Merit, Word2Vec Word Embeddings - 2 March 2018 - rela
------------------------------------------------------------------------------------------------------------------------
ConceptNet5 /related REST endpoint retrieves concepts similar to a concept and ranks them by word2vec word embedding
e Lambda Function Growth algorithm mentioned previously, grows lambda functions for random walks in a text definition
e Gloss Overlap and word-word similarity/relevance is formalized by Neural Tensor Network. ConceptNet /related word2v

Rationale for creating random walks in text definition graph is: Process of Human Text Comprehension is simulated by

References:
----------
513.1 Neural Tensor Network - [Socher-Chen-Manning-Ng] - https://nlp.stanford.edu/pubs/SocherChenManningNg_NIPS2013.p
513.2 The Language Instinct - [Steven Pinker] - Chapter 4 - How Language Works - Word Chains and Chomsky
513.3 Chomsky-Norvig Debate on Algorithmic Versus Statistical Learning - http://daselab.cs.wright.edu/nesy/NeSy13/nor
        <quote>
        - Statistical language models have had engineering success, but that is irrelevant to science.
        - Accurately modeling linguistic facts is just butterfly collecting; what matters in science (and specificall
        - Statistical models are incomprehensible; they provide no insight.
        - Statistical models may provide an accurate simulation of some phenomena, but the simulation is done complet
        - Statistical models have been proven incapable of learning language; therefore language must be innate, so w
        <unquote>
Recursive Gloss Overlap and Recursive Lambda Function Growth are Algorithmic Language Learning Models.
```

513.4 Gold's Theorem - www.lps.uci.edu/~johnsonk/Publications/Johnson.GoldsTheorem.pdf and limit on learnability - l

---

514. (THEORY and FEATURE-DONE) Recursive Lambda Function Growth - ConceptNet5 support, Per random walk
Lambda Function Composition Tree Graph Tensor Neuron Network Intrinsic Merit - 3 March 2018 - related to 515

Recursive Lambda Function Growth implementation has been updated to print per random walk lambda function
composition tree and its Graph Tensor Neuron Network Intrinsic Merit. Similarity has been made configurable by a flag
...
grow_lambda_function3(): Maximum Per Random Walk Graph Tensor Neuron Network Intrinsic Merit : (u'(present(etc,(infra
...

==========

Unsupervised Classification based on top percentile Core numbers of the definition graph(subgraph of WordNet)

==========

This document belongs to class: Mumbai ,core number= 12
This document belongs to class: infrastructure ,core number= 9
...
which is an evidence to the fact intrinsic merit peaks for random walks through high core numbered hub word vertices.

References:

514.1 Cliques and Cavities in Human Brain - [Ann Sizemore, Chad Giusti, Ari Kahn, Richard F. Betzel, Danielle S. Bass
514.2 Algebraic topology and Computational Neuroscience - https://www.technologyreview.com/s/602234/how-the-mathemati

---

515. (FEATURE-DONE) SAT Solver - verbose print and iterations reduced - 4 March 2018

Least Squares LSMR SAT solver implementation has been updated to print LSMR internal calculations. Maximum iterations

---

516. (THEORY) Majority+Voter Composition Hardness Amplification Lemma, Sensitivity Conjecture - Updated Hardness Boun
- related to 318,356 - 5 March 2018

Hardness Amplification for Majority+Voter Composition has been described earlier by adapting XOR Lemma to Majority fu

$$\frac{\text{Hardness of Maj+voter composition}}{\text{Hardness of voter function}} = \frac{[c/\sqrt{n\delta}] + [\text{sum(column2 error entries)}] - [\text{sum(column3 no error}}{\delta}$$

which is based on the sudivided BP* error scenarios matrix mentioned earlier and described again below:

x	f(x) = f(x/e)	f(x) != f(x/e) Noise
x in L, x/e in L	No error	Error
x in L, x/e not in L	Error	No error if f(x)=1,f(x/e)=0 else Error
x not in L, x/e in L	Error	No error if f(x)=0,f(x/e)=1 else Error
x not in L, x/e not in L	No error	Error

x	f(x)	
Randomized Decision tree evaluation	No error	Error

This matrix picturises the various false positives and false negative errors possible in majority voting. Language L

$$\frac{\text{Hardness of Maj+voter composition}}{\text{Hardness of voter function}} = \frac{[\text{sum(column2 entries)}] + [\text{sum(column3 entries)}] - [\text{sum(all no error entries)}}{\delta}$$

But Stability(f(x)) = Pr(f(x)=f(x/e)) - Pr(f(x) != f(x/e)) from definition of Noise stability => Pr(f(x)=f(y)) = (1+S
Sum(column2 entries) = Pr(f(x)=f(y)) = (1+Stability)/2.
Sum(column3 entries) = NoiseSensitivity
=> Hardness of Maj+Voter composition = (1+Stability)/2 + NoiseSensitivity - [sum(all no error entries)]
For majority function, Stability = (2/pi)*delta + O((delta)^1.5) and NoiseSensitivity = O(1/sqrt(n*delta))
=> Hardness of Maj+Voter composition = 1 + (2/pi)*delta + O((delta)^1.5) + O(1/sqrt(n*delta)) - [sum(all no error ent
For large n, previous hardness tends to 1 + (2/pi)*delta + O((delta)^1.5) - [sum(all no error entries)]

From definition of BP*, Probability of No error entries >= 2/3 => Sum(all no error entries) >= 2/3
=> Hardness of Maj+Voter composition <= [pi + 2*delta + k*pi*(delta)^1.5]/2*pi - 2/3
=> Hardness Amplification for Majority+Voter Composition <= [3*pi + 6*delta + 6k*pi*(delta)^1.5 - 2] /[6*pi*delta]

=> Hardness Amplification for Majority+Voter Composition <= 1/delta*(1/2-1/3*pi) + 1/pi + 6k*pi*(delta)^0.5

---

which is huge amplification for small hardness of voter boolean functions.
=> weak voters are hardened by majority
=> Computing majority by a circuit is increasingly hard as voters become weak (intuitively obvious because if voters

```
10993
10994  Sensitivity Conjecture polynomially relates sensitivity and block sensitivity of boolean functions as: s(f) <= bs(f)
10995                              for_all_length_e(Number of correlations x/e for which [f(x) != f(x/e)])
10996         ----------------------------------------------------------------------------------------------------------
10997         for_all_length_e(Number of correlations x/e for which [f(x) == f(x/e)] + Number of correlations x/e for which
10998
10999  Sensitivity and Block Sensitivity are maximum values of e(which can be number of bits or number of blocks) in summati
11000
11001  NOTE: Here hardness of voter boolean function is approximately assumed to equal NoiseSensitivity of Voter Boolean Fun
11002  ----------------------------------------------------------------------------------------------------------------------
11003  => Hardness Amplification for Majority+Voter Composition <= 1/(delta + or - error)*(1/2-1/3*pi) + 1/pi + 6k*pi*(delta
11004  ----------------------------------------------------------------------------------------------------------------------
11005
11006  ----------------------------------------------------------------------------------------------------------------------
11007  517. (THEORY and FEATURE-DONE) Hardness of Majority Voting, SAT Solver Update - Compressed Sensing and Moore-Penrose
11008  - Conflicts - 6 March 2018 - related to 516
11009  ----------------------------------------------------------------------------------------------------------------------
11010  Approximate CNF SAT Solver has been updated to invoke Moore-Penrose Pseudoinverse function to compute approximate inv
11011         (#) Hardness of Majority+Voter Boolean Function Composition in both directions - bottom-up Voter-Majority and
11012         (#) SAT Solver random matrix analysis per-literal probability agrees with observed findings so far in small nu
11013         (#) If Hardness of Majority implies something stronger than P != NP e.g an one-way function or PRG in #P^#P th
11014
11015  ----------------------------------------------------------------------------------------------------------------------
11016  518. (THEORY and FEATURE-DONE) SAT Solver Update - pinv2() - 1000 variables and 1000 clauses - 6 March 2018 - related
11017  ----------------------------------------------------------------------------------------------------------------------
11018  SAT Solver has been updated to use pinv2() pseudoinverse function which has been documented to be faster. After remov
11019  redundant bottleneck, 1000 variables-1000 clauses random SATs have been solved by pinv2() and matrix multiplication t
11020
11021  ----------------------------------------------------------------------------------------------------------------------
11022  519. (THEORY and FEATURE-DONE) SAT Solver Update - 2500 clauses and 2500 variables - LSMR,LSQR and PseudoInverse Benc
11023  - 9 March 2018
11024  ----------------------------------------------------------------------------------------------------------------------
11025  1. SAT Solver class has been changed to accept the algorithm as parameter which can be function names for LSMR,LSQR,P
11026  SOLVE etc.,
11027  2. 2500 variables and 2500 clauses combination of SAT instances have been benchmarked by invoking LSMR, LSQR and Pseu
11028  3. LSMR benchmark has 405 random 3SAT instances and is the fastest, LSQR and PseudoInverse(pinv2) benchmarks have onl
11029  4. LSQR and PseudoInverse(pinv2) are equally slower compared to LSMR by many orders of magnitude.
11030  5. But approximation ratio for LSMR is 91-92% while LSQR and PseudoInverse trend at 96-97% though for small number of
11031  implies an accuracy versus speed tradeoff: LSMR is less accurate but fast while LSQR and PseudoInverse(pinv2) are mor
11032  6. Observed probability per literal for all three coincide with random matrix 1/sqrt(m*n) uniform distribution probab
11033  7. These numbers are only representative figures and number of iterations for LSQR and PseudoInverse(pinv2) are too m
11034  8. Logs have been committed to:
11035    python-src/testlogs/CNFSATSolver.2500clauses2500variables.LSMR.log.9March2018
11036    python-src/testlogs/CNFSATSolver.2500clauses2500variables.LSQR.log.9March2018
11037    python-src/testlogs/CNFSATSolver.2500clauses2500variables.PseudoInverse.log.9March2018
11038
11039  ----------------------------------------------------------------------------------------------------------------------
11040  520. (THEORY and FEATURE-DONE) SAT Solver Update - Non-Uniform Choice - For both equal and unequal number of clauses
11041  related to 519 - 11 March 2018
11042  ----------------------------------------------------------------------------------------------------------------------
11043  SAT Solver has been updated to include a new Non-Uniform Choice function for creating random SAT instances by choosin
11044  probability = 1/sqrt(m*n) which works for both equal and unequal number of clauses. This new function chooses non-uni
11045          1000 variables, 1000 clauses - CNFSATSolver.1000variables1000clauses.NonUniformChoice.log.11March2018
11046          1100 variables, 1200 clauses - CNFSATSolver.1100variables1200clauses.NonUniformChoice.log.11March2018
11047          1200 variables, 1100 clauses - CNFSATSolver.1200variables1100clauses.NonUniformChoice.log.11March2018
11048  From logs it is evident non-uniform choice function defined as previously approximately simulates the bias and almost
11049
11050  ----------------------------------------------------------------------------------------------------------------------
11051  521. (THEORY and FEATURE-DONE) SAT Solver Update - Non-Uniform Choice - 3200 variables and 3100 clauses and Alpha=4.2
11052  - 12 March 2018
11053  ----------------------------------------------------------------------------------------------------------------------
11054  SAT Solver has been updated for some aesthetics - literal selection in non-uniform choice has been parametrized: sequ
11055   based on how three random literals are chosen per random 3SAT clause - sequentially as 3*nP1 or simultaneously as nP
11056  observed between these two choices. Number of variables is set to 3200 and clauses to 3100. LSMR iteration has been i
11057  reduced to 10. For few random 3SATs, observed MAXSAT approximation ratio average stands at ~95%. This accuracy has be
11058  where a phase transition occurs from easy-to-hard. Number of unsatisfiable formulae increase for alpha > 4.26. MAXSAT
11059
11060  References:
11061  -----------
11062  521.1 SAT Solvers and Phase Transition at Alpha=4.26 [Clause/Variable Ratio] - [Carla P. Gomes, Henry Kautz, Ashish S
11063
11064  ----------------------------------------------------------------------------------------------------------------------
11065  522. (THEORY and FEATURE-DONE) SAT Solver Update - Non-uniform Choice 2 - 5000 variables and Alpha=4.267 and some int
11066  - 15 March 2018
11067  ----------------------------------------------------------------------------------------------------------------------
11068  SAT Solver has been updated for a new nonuniform choice function nonuniform_choice2() which has following algorithm:
11069          - Each permutation nPn of length n is elongated to n*n*alpha length array by replicating a non-variable X of
11070          - Literals are chosen from this new array as (n*n*alpha)P1 or (n*n*alpha)P3 in a while loop till a valid lite
11071          - This creates a probability of fraction n/(n*n*alpha)=1/(n*alpha) for per literal choice.
11072  But this implementation was found to be no different from nonuniform_choice() which chooses a submatrix in per litera
11073
```

```
Here the intuition on how LSMR/LSQR works has to be mooted - Each binary assignment string can be thought of as a ste
                        ----  -
                        |  | |
                       ---  ---
where troughs represent 0s and peaks 1s. LSMR/LSQR finds a set of real valued points on a sinusoidal polynomial which

Probability of an LSMR/LSQR real-to-binary round off assignment failing to satisfy a random 3SAT:
-----------------------------------------------------------------------------------------
For each binary variable xi, LSMR/LSQR creates a real value which is rounded off as:
       xi   -    0  if  xi < 0.5
       xi   -    1  if  xi > 0.5
This round off can fail if:
       xi   -    1  if  xi < 0.5
       xi   -    0  if  xi > 0.5

Lovasz Local Lemma Analyses described earlier further explain the clause-variable dependency graph(Factor graph) scen

-----------------------------------------------------------------------------------------------------------------------
523. (THEORY and FEATURE-DONE) SAT Solver Update - Variables and Alpha as parameters and two Alpha Versus MAXSAT rati
related to 522 - 16 March 2018
-----------------------------------------------------------------------------------------------------------------------
SAT Solver has been updated to accept commandline parameters of number of variables and Alpha (Clause/Variable ratio)
               $python CNFSATSolver.py <number_of_variables> <alpha>
Some debug prints have been changed. Following sets of SATs were solved and variation of MAXSAT approximation ratio v
captured as graph (for nonuniform_choice()):
-------------------------------------------------------------
Number of Variables - 600 (after atleast 15 random 3SAT iterations)
-------------------------------------------------------------
Alpha                          Observed MAXSAT Approximation Ratio
-------                        ----------------------------------
1                              94.5%
2                              92.7%
3                              91.5%
4                              90.74%
4.267                          90.45%
5                              90.28%
6                              90.25%
-------------------------------------------------------------
Number of Variables - 300 (after atleast 15 random 3SAT iterations)
-------------------------------------------------------------
Alpha                          Observed MAXSAT Approximation Ratio
------                         ----------------------------------
1                              95%
2                              93%
3                              92%
4                              90.6%
4.267                          90.6%
5                              90.49%
6                              90.73%

Logs for an iteration of 5000 variables and Alpha=4.267 has been committed to python-src/testlogs/CNFSATSolver.5000va
-----------------------------------------------------------------------------------------------
SAT Solver has been updated for a new nonuniform_choice3() function based on an Epsilon Bias Test Snippet EpsilonBias

-----------------------------------------------------------------------------------------------------------------------
524. (THEORY and FEATURE-DONE) SAT Solver Update - nonuniform_choice3() updated for constant multiple in expansion of
- 19 March 2018 - related to 523
-----------------------------------------------------------------------------------------------------------------------
1. CNFSATSolver.py nonuniform_choice3() has been changed to expand the variables array by following relation:
        n + x = n*sqrt(alpha)
        x = n*(sqrt(alpha)-1)
        => array of n variables is expanded to array of size n*(sqrt(alpha)-damp) for alpha = number_of_clauses/numbe
2. damp variable has been hardcoded to 2 than 1 which approximates theoretical 1/sqrt(m*n) probability well because c
3. This expansion creates an approximate nonuniform probability 1/n*sqrt(alpha) per literal excluding replicated skew
filtered in the while loops
4. Example SAT Solver logs for 1000 variables and Alpha=4.267 has been committed to python-src/testlogs/CNFSATSolver.
5. MAXSAT Approximation ratio again is 90-91% for 20+ random 3SATs and nonuniform per literal probability almost matc
(till penultimate decimal)

-----------------------------------------------------------------------------------------------------------------------
525. (THEORY) Counterexample in Majority Voting, Timeline Evolution of Belief Propagation, Intrinsic Merit - related
-----------------------------------------------------------------------------------------------------------------------
Previous sections described a counterexample in Majority Voting for a question:
       "Did an artefact exist millenia ago?"
There are two possible answers: Yes and No which depend on intrinsically determining truth of answer to this question

-----------------------------------------------------------------------------------------------------------------------
526. (THEORY and FEATURE-DONE) Hardy-Ramanujan Approximate Ray Shooting Queries for Factors - Optimization in Computa
Factorization - 21 March 2018 - related to 487
-----------------------------------------------------------------------------------------------------------------------
1. Ray Shooting Queries based on Hardy-Ramanujan Theorem for Normal Order of number of factors of an integer has been
```

```
new function.
2. Each approximate factor is queried by a ray from origin of slope tan(m*pi/(2*k*loglogN)) intersecting hyperbolic a
               sqrt(N/[tan(m*pi/(2*k*loglogN))])-1 for m=1,2,3,...,kloglogN
3. Two integers have been factorized using local tile search and comparison between approximate factors and actual fa
       python-src/testlogs/DiscreteHyperbolicFactorizationUpperbound_TileSearch_Optimized.HardyRamanujanRayShootingQ
       python-src/testlogs/DiscreteHyperbolicFactorizationUpperbound_TileSearch_Optimized.HardyRamanujanRayShootingQ
4. Approximate Ray Shooting could be useful for integers having large value of factor multiplicity BigOmega (sum of p
5. Constant k has been hardcoded presently and has to be heuristically found.
6. Approximate Factors are helpful for sieving huge integers and binary search can be localized based on these approx

------------------------------------------------------------------------------------------------------------------------
527. (FEATURE-DONE) Unsorted Search Update - Streaming Abstract Generator File Datasource Support, Prefix-Suffix subs
- 23 March 2018
------------------------------------------------------------------------------------------------------------------------
1. Hardcoded File name in Streaming Abstract Generator has been replaced by a file name variable and file contents ar
stripped of leading and trailing whitespaces and yielded.
2. Padding of '#' in Unsorted Search has been replaced by '0'. Revised Unsorted Search Algorithm in https://github.co
3.New function to print contents of all hashtables has been added.
4.Function create_prefix_suffix_hashtables() initializes prefix and suffix hashtables for strings from Streaming Abst
5.search_number() function has been rewritten to search for all prefixes and suffixes hashtables and match True/False
6.New input file First100Primes.txt has been created.
7.Primes and Non-Primes are lookedup and logs has been committed to testlogs/UnsortedSearch.log.23March2018

------------------------------------------------------------------------------------------------------------------------
528. (THEORY) Consensus, Pareto Optimality, Intrinsic-Perceived Value/Merit Equilibrium, Fair Division, Nultiple Ager
------------------------------------------------------------------------------------------------------------------------
Intrinsic Fitness/Merit and Perceived Merit Equilibrium has been described earlier as an approximate consensus measur
merit of an entity. Consensus problem for absolute intrinsic merit arises in the following context of Fair Division w
to be allocated amongst multiple agents fairly (also known as MARA-Multi Agent Resource Allocation). Fairness in allo
For a utility function u:(Good,Agent)->Happiness, set of agents (x1,x2,x3,...,xn) and goods/services/resources (r1,r2
Most obvious application of Pareto Optimality is Pricing or Determining Value of Goods. Allocation is Envy-Free if no

Traditional Ranking methodology followed is to compute ranking scores per website URL independent of other URLs i.e e

References:
-----------
528.1 MARA Survey - https://staff.science.uva.nl/u.endriss/MARA/mara-survey.pdf

------------------------------------------------------------------------------------------------------------------------
529. (THEORY and FEATURE-DONE)  Recursive Lambda Function Growth - Graph Tensor Neuron Network Intrinsic Merit for Ra
- Analysis for different text - 26 March 2018
------------------------------------------------------------------------------------------------------------------------
1. Text input for Recursive Lambda Function Growth has been updated
2. Graph Tensor Neuron Network Intrinsic Merit for Lambda Function Composition of Random Walks of Definition Graph ha
log testlogs/RecursiveLambdaFunctionGrowth.log.GraphTensorNeuronNetwork.26March2018
3. Maximum Graph Tensor Neuron Network Intrinsic Merit occurs for Random Walk Composition:
grow_lambda_function3(): Maximum Per Random Walk Graph Tensor Neuron Network Intrinsic Merit :
(u'(housing(protective,(certain(zone,(something(target,(part(include,(urban(area,(Chennai(city,formerly))))))))))))',
4.Top percentile Unsupervised Classes of this text by Dense Subgraph Discovery (Core Numbers) are:
========================================================================================================
Unsupervised Classification based on top percentile Core numbers of the definition graph(subgraph of WordNet)
========================================================================================================
This document belongs to class: arsenic ,core number= 16
This document belongs to class: Greenwich_Village ,core number= 14
This document belongs to class: state_of_matter ,core number= 14
This document belongs to class: order ,core number= 14
This document belongs to class: include ,core number= 10
This document belongs to class: part ,core number= 10
This document belongs to class: exploitation ,core number= 9
This document belongs to class: area ,core number= 9
This document belongs to class: three ,core number= 9
This document belongs to class: collector ,core number= 8
This document belongs to class: housing ,core number= 8
This document belongs to class: January ,core number= 8
This document belongs to class: trey ,core number= 8
This document belongs to class: Chennai ,core number= 7
This document belongs to class: urban ,core number= 7
This document belongs to class: planning ,core number= 6
This document belongs to class: zone ,core number= 6
This document belongs to class: free-base ,core number= 6
This document belongs to class: travel ,core number= 6
This document belongs to class: one ,core number= 6
This document belongs to class: target ,core number= 6
This document belongs to class: something ,core number= 6
This document belongs to class: republic ,core number= 6
This document belongs to class: government ,core number= 5
This document belongs to class: legal_power ,core number= 5
------------------------------------------------------------------
which reasonably coincide with the vertices of the maximum intrinsic merit random walk and capture the Legal/Gubernat
of the text(though there are WordNet anomalies printing 'arsenic' etc., as classes. These anomalies are caused becaus
```

```
Essence:
--------
Meaning of a text is approximated as:
        - Create a Definition Graph Representation of Text from some Ontology
        - Do random walks/Hamiltonian on the graph to simulate the human text comprehension
        - Get classes of the text from Recursive Gloss Overlap Unsupervised Classifier
        - Compute Lambda Composition Trees for all random walks in definition graph
        - These random walk lambda composition trees are approximations of all possible meanings of the text
        - Compute Graph Tensor Neuron Network Intrinsic Merit for all random walk lambda composition trees
        - Lambda Composition Tree of Maximum Graph Tensor Neuron Network Intrinsic Merit is the most likely approxima
        - This is obvious because this maximum merit tree has Neuron Tensor Network Relations of maximum similarity/t
these similarities are composed and belief propagated

-----------------------------------------------------------------------------------------------------------------
530. (FEATURE-DONE) Recursive Lambda Function Growth - Simple Cycles and Rich Club Coefficient - 27 March 2018
-----------------------------------------------------------------------------------------------------------------
1.Recursive Lambda Function Growth implementation has been updated to loop through cycles in the definition graph of
between Cycles and Random Walks by a boolean flag ClosedPaths.
2.Closed Paths or Cycles simulate the meaning better and the tree lambda composition obtained from cycle vertices has
found to approximate meaning more closely.
3.Logs for this has been committed to testlogs/RecursiveLambdaFunctionGrowth.log.GraphTensorNeuronNetwork.27March2018
4.Rich Club Coefficients of the Definition Graph has been printed for each degree as a measure of connectivity of hig
Rich Club Coefficient of the Recursive Gloss Overlap Definition Graph: {0: 0.004865591072487624, 1: 0.016526610644257
5.Top core number classes from Unsupervised Recursive Gloss Overlap Classifier:
=================================================================================================
Unsupervised Classification based on top percentile Core numbers of the definition graph(subgraph of WordNet)
=================================================================================================
This document belongs to class: incorporate ,core number= 4
This document belongs to class: environment ,core number= 4
This document belongs to class: regional ,core number= 4
This document belongs to class: `in ,core number= 4
This document belongs to class: component ,core number= 4
This document belongs to class: exploitation ,core number= 4
This document belongs to class: useful ,core number= 4
This document belongs to class: relating ,core number= 4
This document belongs to class: unit ,core number= 4
This document belongs to class: particular ,core number= 4
This document belongs to class: making ,core number= 4
This document belongs to class: process ,core number= 4
This document belongs to class: sphere ,core number= 4
This document belongs to class: something ,core number= 4
This document belongs to class: goal ,core number= 4
This document belongs to class: farm ,core number= 4
This document belongs to class: urbanization ,core number= 4
This document belongs to class: strategic ,core number= 4
This document belongs to class: ' ,core number= 4
This document belongs to class: increase ,core number= 4
This document belongs to class: comforts ,core number= 4
This document belongs to class: city ,core number= 4
This document belongs to class: district ,core number= 4
This document belongs to class: farming ,core number= 4
This document belongs to class: urban ,core number= 4
This document belongs to class: way ,core number= 4
This document belongs to class: plan ,core number= 4
This document belongs to class: do ,core number= 4
This document belongs to class: contain ,core number= 4
This document belongs to class: see ,core number= 4
This document belongs to class: state ,core number= 4
This document belongs to class: region ,core number= 4
This document belongs to class: proposal ,core number= 4
This document belongs to class: life ,core number= 4
This document belongs to class: decisiveness ,core number= 4
This document belongs to class: lives ,core number= 4
This document belongs to class: metropolitan ,core number= 4
-------------------------------------------------------------------------------------
coincide reasonably well to the Maximum Merit Cycle as below and the meaning of the text can be inferred from the com
-------------------------------------------------------------------------------------
Cycle : [u'particular', u'regional', u'region', u'something', u'see', u'make', u'plan', u'goal', u'state', u'city', u
Cycle Composition Tree for this cycle : (u'particular((region(regional,(see(something,(plan(make,(state(goal,(urban(c
maximum_per_random_walk_graph_tensor_neuron_network_intrinsic_merit= (u'(regional(particular,(`in(region,(pronounce(w
=====================================================================
grow_lambda_function3(): Graph Tensor Neuron Network Intrinsic Merit for this text: 3804.34147246
6.NetworkX library has been upgraded to recently released version 2.1

-----------------------------------------------------------------------------------------------------------------
531. (THEORY-FEATURE-DONE) Least Squares SAT Solver Benchmarks - 1000 variables - Non-uniform choice 3 - different va
-----------------------------------------------------------------------------------------------------------------
1. SAT Solver (LSMR) has been benchmarked for 1000 variables and for varying values of Alpha as below. MAXSAT approxi
atleast 20 iterations of random 3SAT for nonuniform_choice3() are:
Alpha = 1 - Observed - MAXSAT-APPROXIMATION Ratio - Moving Average Percentage of Clauses per CNF satisfied so far: 94
Alpha = 2 - Observed - MAXSAT-APPROXIMATION Ratio - Moving Average Percentage of Clauses per CNF satisfied so far: 92
```

```
Alpha = 3 - Observed - MAXSAT-APPROXIMATION Ratio - Moving Average Percentage of Clauses per CNF satisfied so far: 91
Alpha = 4.267 - Observed - MAXSAT-APPROXIMATION Ratio - Moving Average Percentage of Clauses per CNF satisfied so far
Alpha = 5 - Observed - MAXSAT-APPROXIMATION Ratio - Moving Average Percentage of Clauses per CNF satisfied so far: 89
Alpha = 6 - Observed - MAXSAT-APPROXIMATION Ratio - Moving Average Percentage of Clauses per CNF satisfied so far: 89
Alpha = 7 - Observed - MAXSAT-APPROXIMATION Ratio - Moving Average Percentage of Clauses per CNF satisfied so far: 89

2.As usual for equal variable-clauses MAXSAT ratio is the maximum at 94-95% and decreases for increasing alpha. For A

3.Log excerpts have been captured in testlogs/CNFSATSolver.1000variablesDifferentAlphasBenchmarks.29March2018

-----------------------------------------------------------------------------------------------------------
532. (THEORY) Packing/Filling/Tiling problems, Complement Functions/Diophantine Equations, Ramsey Coloring,
Exact Cover, Matiyasevich-Robinson-Davis-Putnam Theorem  - related to 462,463,491 - 19 April 2018
-----------------------------------------------------------------------------------------------------------
As mentioned in previous sections, complement functions are subsets of set of Diophantine Equations defining
each diophantine set in an exact cover - this generalizes definition of complement functions to exact cover of size g

Reference:
----------
532.1 Pentominoes Tiling Exact Cover for Chess board - https://en.wikipedia.org/wiki/Exact_cover - Each pentomino til
532.2 Euclidean Ramsey Theory - http://www.math.ucsd.edu/~ronspubs/pre_Euclidean.pdf - [RL Graham] - Euclidean Ramsey

-----------------------------------------------------------------------------------------------------------
533. (THEORY and FEATURE-DONE) Complement Diophantine and Factorization - Pell Equation Solver Update -
20 April 2018
-----------------------------------------------------------------------------------------------------------
1. Complement Function Implementation has been updated to invoke sympy Pell Equation solver for some N and D
in x^2 - D*y^2 = N
2. Factorization reduces to integer solutions of Pell Equation but the converse is harder - factors must be known
a priori as pq = N from some factoring algorithm. Then following equations can be solved:
        x + sqrt(D)*y = p
        x - sqrt(D)*y = q
for x and y.

log excerpts:
==========================================================================
Pell Diophantine Equation - Factorization reduces to solving Pell's Equation (for some N)
==========================================================================
Solution for Pell Equation : x^2 -  91 *y^2 =  1 :
( 1574  + sqrt( 1 ))/ 165  * ( 1574  - sqrt( 1 ))/ 165  =  91

-----------------------------------------------------------------------------------------------------------
534. (THEORY and FEATURE-DONE) Complement Diophantine and Factorization - Pell Equation Solver Update 2 -
20 April 2018
-----------------------------------------------------------------------------------------------------------
1. Complement Function Implementation has been updated to invoke sympy Pell Equation solver for some N and D
in x^2 - D*y^2 = N for D=1
2. Following are the factors p,q of N:
  x + y = p
  x - y = q
for Pell equation solutions x and y. This is exponential in number of bits of N and equivalent to most
Number Field Sieve algorithms for Factorization in vogue. Computational Geometric Factorization in NC
implies x and y can be found in parallel polynomial time (which is quite surprising given the antiquity and
notorious difficulty of this ~1500 year old problem).
3. Pell's Equation for Factoring and Prime polynomials (e.g Matiyasevich, Jones-Sato-Wada-Wiens) are thus
complement diophantines representing the exact cover {Primes,Composites} of the Set of Natural Numbers

log excerpts:
==========================================================================
Pell Diophantine Equation - Factorization reduces to solving Pell's Equation (for D=1 and some N)
==========================================================================
Solution for Pell Equation : x^2 -  1 *y^2 =  989295 :
( 1084  +  431 ) * ( 1084  -  431 ) =  989295

-----------------------------------------------------------------------------------------------------------
535. (THEORY) Complement Functions, Complement Graphs, Ramsey coloring, Intrinsic Merit of Text-Graph -
related to 2,338 - 21 April 2018
-----------------------------------------------------------------------------------------------------------
Relations between the concept of complement functions and complement graphs have been described earlier in the
context of Perfect Graph Theorem. This notion of complement graphs has direct application in graph representation of

There is a known theorem which states that if a graph G is disconnected its complement G's is connected and vice-vers

References:
----------
535.1 Introduction to Graph Theory - [Douglas B.West] - Perfect Graph Theorem for complement graphs, Ramsey Theorem f

-----------------------------------------------------------------------------------------------------------
536. (FEATURE-DONE) ConceptNet Client Upgrade to 5.6 - REST endpoints for Emoticons - 23 April 2018
-----------------------------------------------------------------------------------------------------------
1. ConceptNet Client has been updated for new REST endpoints in ConceptNet 5.6
```

```
2. Function for querying emotions has been added for new emoji endpoints in ConceptNet 5.6
3. logs for this have been committed to testlogs/ConceptNet5Client.log.23April2018

-----------------------------------------------------------------------------------------------------
537. (THEORY) Shell Turing Machines, Word2Vec and Intrinsic Merit - 24 April 2018 - related to 42
-----------------------------------------------------------------------------------------------------
Shell Turing Machines described earlier are experimental enhancements to Turing Machine definition which
introduces dimension as a parameter in addition to tapes, alphabets, head etc., Adding dimension to Turing
Machines has immense applications and simulates lot of real world computations which span across multiple
dimensions. For example a Turing Machine defined in dimension d+1 has more computational power than a machine
defined in dimension d i.e L(T(d)) is in L(T(d+1)). This is a dimensional hierarchy as opposed to Time and Space
hierarchies of Turing Machines. This has striking applications in graph representation of texts. Each word
vertex in a definition graph G1 can be represented as a vector in a space of dimension d. Word2Vec embeddings
(e.g ConceptNet) already implement this by mapping each word to a vector in space of some dimension.
Then another definition graph G2 of words defined on a vector space of dimension d+1 can elicit more meaning
from text than G1. Recursive Lambda Composition Trees for G1 and G2 are two Turing-equivalent computation models
 approximating the meaning - G2 approximates better than G1. In other words intrinsic merit computed for G2 is
more accurate than G1 for same text. Present algorithms for Recursive Gloss Overlap and Recursive Lambda
Function Growth do not affix dimension information for word vertices.
```

About (/about)
Site Status (/blog/category/sitestatus/)
@sfnet_ops (https://twitter.com/sfnet_ops)
Create a Project (/create)
Software Directory (/directory/)
Top Downloaded Projects (/top)
Blog (/blog/)
@sourceforge (https://twitter.com/sourceforge)
Resources (https://library.slashdotmedia.com/)
Articles (/articles/)
Site Documentation (https://p.sf.net/sourceforge/docs)
Support Request (/support)
(/)

© 2018 Slashdot Media. All Rights Reserved.

Terms (http://slashdotmedia.com/terms-of-use) Privacy (http://slashdotmedia.com/privacy-statement/) Opt Out (http://slashdotmedia.com/opt-out-choices) Advertise (http://slashdotmedia.com/)