What is computer science:

Problem solving, using computer in the most efficient way(with the lowest complexity)

Complexity: time (reduce duration cost for cpu/gpu)

Space(big data aera, cancer center: 1 saan 2G+) O(n)

Kolmogorov: founder of axiomic probability theory/chaitin

k-complexity

reduce Size/length of the alg (algorithm)

dry: don’t repeat yourself

Efficiency: achieve good time + space complexity

1 concept: important In cs/se

2.alg design/code

3.math/stat/probability analysis

language: matlab

good in python: similar as matlab

low Kolmogorov

understandability: for peer programmers to know your idea form your code(your src code)

how to achieve it: clean logic , variable/function name meaningful, add comments (every function have a paragraph, every 10 lines, add a sentence help other or self to understand)

mac:

OS X: variant of unix

More beautiful

User-friendly: to customer (not coders) easy and fun to use

Don’t be proud of making system ugly (like unix ..)

A/B testing: abused by IT/ google

Has 2 version, randomly choose IP/customer, see which version is more popular

Ask paid tester to do:#limit

Big data: size of data to be processed is huge, O(n) space

chatGPT: 500GB RAM

big company: n customers ( n is huge) indexed from 0 to n-1 which means range (n)

free food to every customer, no double dipping all allowed, no one double dipping

suppose only 1 customer didn’t pick up free food yet

how to find out who the customer is ?

efficient design:

LUT: remember the index get ride of LUT

N entries

1 number to single out the missing index/person/customer

Afterwards, sort the LUT

Space complexity: O(n) not viable

Reduce space from O(n) to sublinear, O(1)

Math to solve this:

Sum total of those who already came

Sum from 1 to n: n\*(n+1)/2 = F\_sum

F\_sum – sum: will be the one (index) never shows up

What if 2 customers didn’t come?

F-sum –sum =I +j

100: sum of the 2 index

2 unknows, 2 equations to uniquely resolve I ,j

i+j=F-sum-sum (1)

i\*\*2+j\*\*2= fixed diff value

…

i\*\*k+j\*\*k = fixed value’

stream data proc/big proc

OS: chunk swapping : garbage collection

Allocate space

Request a new (huge) array to save 1-5, 45123 to the original location

O(n) space: sub linear to n

Using algorithm to simulate circular linked list

Trade time for space

Program = data structure + algorithm

Binary search: log(n): almost linear, hashing table: O(1) search

Space for hashing O(n)

***9/5***

chunk swapping: array as ds, alg to emulate the linked list ds: data structure

simple ds , smart alg

1 2 3 4 5 -> 4 5 1 2 3 1 2 3 4 5 -> 3 4 5 1 2

k: 1 to n-1

k\*n: time

k<n/2: shift to the left

k>n/2: shift to the right

n-k: <n/2

min(k, n-k): improved the efficiency

solve problem: with lowest complexity

chunk swapping: array , use loops to emulate circular linked list

Ds: data structure

Alg: algorithm

program(solve) = data structure + algorithm

solve #1: smart alg, simple ds

solve #2: smart ds, simple alg

efficiency: time + space complexity

trade-off: both low on time and low on space(big breakthru)

high on both: bad programming

better

linked list:

change Head, Tail, break one link to original head

time complexity is much better than alg .1

choose the ds wisely, alg is so simple

alg 2 better than alg.1

software engineering: motivation

system based on computer: code + fix (software develops)

write the code, deliver system, if trouble surfaces, fix it

DoD US: fund $$, projects

Contractor lifeline is DoD

Software crisis: let drop all software projects (…)

Expensive ($1M, $3M, $5M) ->$15M

Unreliable: today works, tomorrow not working, fix

Hard to maintain

Software engineering: apply successful engineering principles to software develop

Default: agile model: variant of water-fall model where the system in cut into small pieces

How to build a bridge?

(similar to water-fall model). (first and dominant for years, not popular any more)

1. Meetings: feasibility studies, field study, secure $
2. Form a blueprint (design): by engineer, more meetings to evaluation design
3. Send construction engineers to build
4. Testing phase: last a while
5. Deliver & walk/drive
6. Maintenance
7. Retire/blow up

Software: copy cat of this engineer principle

Life cycle models: the series of steps to develop a huge software system.

Difference between software system and ord/ conventional engineering projects:

Have many visual tools to help communications between clients/customer and developer

Visualization (can’t see /touch directly)

Moving targets: 100m, 400m … marathon

Clients/bosses feel entitled to ask for changes lack of mutual understanding

Say absolutely no: allow some meeting, justify why not

Use many diagrams: finite state machine (FSM), petri-nets, e-r diagram, class diagrams

Have diagrams and equations

Natural language: ambiguities

***9/7/2030***

life cycle model: agile model: many meetings daily/weekly

how frequency stand up, sprints

M. Hamilton/2016, got solve real world problem using lowest complexity

Time/length/space

Try to create own library

Own library: cheating (chatGPT: collect by others’ work)

CASE: computer aided software engineering

Any tools that can help improve the software engineering process

Case: can help us in coding it useful and powerful (google/Facebook has own case Tensorflow: google brain release for free 2nd most popularly used DL/ML pack

Facebook/meta: pytorch torch 1st popular esp in academia

Try new idea

Git: case to collaborate , version control ,regret, change back

Make: use case.

Huge system: selective compiling

Makefile: dependencies tree, compile those changed ones,

Editor: CASE tool,

Notepad? X

Python: vs code :

PyCharm (out of box, space hungry)

Sublime: used to be popular

Vim/meovim:config . vimrc as good as vs code

Local: gui based editor

Vim/neovim: server side coding

Vi: original unix editor, git

shell script: automate

windows: batch .bat (ugly by ms) PowerShell: big deal object based shell programming

general ones

CS/CE: more popular Just coder

Life cycle: coding is small part of life as se

Math modeling: good in math, calculus (Deep Neural network: DNN CNN: convolutional NN) biology

Taylor series, chain rule: f(g(x)) : f’\*g’

Gradient descent to change parameters of the NN

Aircraft: nothing to do with bird

Google: most $$ svd: eigen vector

Singular vector decomposition

Matrix, eigen analysis links,

M=USV^T (svd)

Probability: math derivative/statistics: inductive evidence

Log(n) and n\*\*0.0001

Alg 1 : log(x)

Alg 2: x\*\*0.0001

Limit x->+oo log(x)/x\*\*d = 0

Log(x) is cheaper than x\*\*d, when x ->oo

Log(x)’ =1/x

X\*\*d’=x\*\*(d-1)

Sympy ( solve math problem )

Skip list: data structure in big data

Express line of subway:

CCNY -> time square

Smart way to go:

First set a objective function: take the least possible stops

# of stops

best/smartest plan should be with least number of stops

average # of stops all nyers will make ; grand average n stations

n/2: math assumption, roughly true

introduce express line: reduce avg # of stops nyers will make optimal # of express stops:

find optimal x which can minimize objective function(# of avg stops nyers

1 too few, N is too many

In US military: military: engineering, not (art)

Engineering is the key to the success of us

von Neumann: father of CS , mergesort

decimal #: 10 states

binary:

computer can beat human chess champ in 10 years

9/12/2023

express line of subway: foundation for skip list

NYC: ccny ce: blueprint of subway of nyc: classified, national

Construct an express line: $$

Justify the extra $$ to build//dev/maintain

NYers: pay $$

Feasibility study: justify/model

Avg: average

Local only: avg nyers N/2 # of stops you have to make

Obj: as few stops as possible

Add express, prove avg stops avg nyer: no favor or disfavor to anyone

Too many: no reduction of average stops

Too few: favor small group of people unfair

>1 <N, X :unknown , try to find the math behind to optimize # of stops by choose the right x

find optimal X( # of express stations)

assume we added express line with x stops;

avg # of stops nyers will make:

have some simplification

calculus: continuous, 1st smooth, f’(x) should exist

assume curve/surface having 1st serivative

single (direction) linked list

avg # of exp stops avg nyers will make? X/2

avg # of local stops avg (reasonable) nyers will make? X/N/2

objective: total # of stops avg nyers will make;

f(x) = x+N/x

N is known, x is unknown, find x s.t. f(x) is minimized

X= arg\_min\_x f(x)=argmin\_x(x+N/x), N is a constant

f’(x) = (x + N/x)’ = 1-N/x\*\*2 =0

x = sqrt(N)

if 100 local stops: exp : 10

O(N) -> O(sqrt(N)) sublinear, pay$$

Fun to consider: doubline link list;

Construct reasonable objectives

Make math simplification to facilitate analysis, function to be opt

Optimize: numerically solve: torch/tensorflow: black box/machine

To find parameters to opt objective

Statistics: CS

P(people walk thou the wall) = 0.0000000000000000001

CS -> Boolean logic

math modeling:

a group of people, each person has a annual income x

what is the typical annual income of this group?

Best statistic to represent the annual income of this groups

Avg/mean value: typical annual income (?) one alternative

Median value: sort array, take the middle, odd: even: average of the middle 2

Which is better?

Mean value: not robust, over-sensitive to p/n outliers

Classic stats, result from Normal (Gaussian) distribution

Justify why mean is bad and median is more robust

Setting up an objective

Y I’s, i=1….n

Find a stat A, such that overall error will be minimized

Gauss:

F(x) (total error of using x to represent Yi’s)

(Yi –x )\*\*2

f(x)=

mean value minimize the mean square error (MSE)

however median is better, math is uglier

CLT: central limit throrem

Data not that large, not that unrelated

Measured value = real value + Gaussian(GLT)

Robust stats: order stats are more robust

Median , quantile

Median: outliers’ contributions will be discarded

If n is small: enumerate

If n is not small: need statistic

ML,DL: different objective

Form the objective (reasonable);

9/14

mean is the stat minimizes the MSE, so sensitive to outlier ()\*\*2

Gauss: math is elegant

Median is more robust from gauss’s diary

Non-E geometry,

FFT: fast Fourier transform (transform)

1965: FT: O(n\*\*2)

Tukey and ??: robust stats: merge sort like alg, O(nlogn) mp3,jpeg,mpg: based FFT

Median: more robust, small # of outliers won’t affect the median 100k, 100bil

Mse f(x) = \sum (x-xi)\*\*2

L1 error (Manhattan):\sum |x-xi|: outlier’s contributed Is not as serious

4-color thm: planar map: use 4 colors to color no adjacent regions share the color

Measure error:

Euclidean distance L\_2

Manhattan distance L\_1: for reasonable NYers to move from one position to another position

Use cal 1 to find out the x minimizing the sum of L1 errors;

The median value by def will make the sum total to the derivative = 0

Error measure: positive (allow negative: cancel)

A pond, unknown # of fish: ask to estimate # of fish as close as possible

Population to be evaluate is too large to enumerate

Sampling: alive, freely moving, fish, tiger, bear

Method: capture 10 fish/bear/tiger -> put a label/electronic tag to them, put them back ->

Recapture 10 fish/bear/tiger: count # of tagged alive animals

For example: 2 of them are tagged, # of fish this pond: fraction sampling (method used in most stem disciplines) 50

Have the methodology to justify: statistical reasoning

Educated guess of

Mobile population: fraction sampling

Static population: huge lawn, how many grasses we have?

Mountain, # of trees we have

Estimate of (have stats, cannot enumerate)

10 random throwing

test the goodness of a fertilizer (which method is better)

try 2 brands in 2 different years, Yield 1 yield2, whichever is larger will win 2 years are not fair condition: weather, based on lazy/hardworking the farmer no control over the fertilizer

cancel all factors except the fertilizer

revolutionize the way to do science& engineer

randomization: in agriculture

medicine: double-blind testing is behind the huge program in medicine

testing: to decide the goodness/badness of an approach

cut the field into small plots, toss a coin, head: use A in this plot, tail: use B ensure the farmers have no idea which plot received which fertilizer (famers’ contribution is nil)

farmers: cancel

weather: cancel

company (A/B): no control chance det the app of different b rands

geography: randomly assigned, cancel each other since we have lots of plots

scientific testing method

double –blind testing:

separate people into 2 groups: one testing (take the new med) one control (take sugar)

randomized no one can control:

volunteer no idea belong to t or c group?

Lie: ask for bribe: behave diff affect result

new med/vaccine: doctor should not decide which person will receive the med

blind to doc/vol: safeguard the fairness of the test

***9/19***

double-blind:

a fraction of papers/ proposal evaluation

remove author’s names after publication

author should not know who will be the evaluator

single-blind:

put authors/proposers name

who will evaluate your paper/proposal

double-blind: more objective

single-blind: favor famous guys/instance

cs: google search engine ,vs bing vs duckduck go compare three item

: privacy:

why do you choose google?

Google return more related results/ have you objectively evaluation?

Right way: magic word: double blind:

Keyword should be neutral: a long list of words disinterested to all search engine

No know which engine you are using : randomly choose which engine to use

Evaluation person: have no idea of google/bing/duckduckgo

Gold standard to tell apart science and non-science

Testing to falsify claims

Political science: no result

Philosophers: plato better than conf

Science decide: order

Math is not science: a tool/ man made art

Provide hard evidence: scientific way to evaluate search?

Quick sort: choose an anchor, figure out the final position of anchor

How to choose anchor: first element: vulnerable to attacks

Game theory: open to vicious attacks

Randomized algorithm: solve shark

Ccny cs student vs hunter college cs student:

How to eval:

Randomized repeat this process several times

Binary search O(log n)

Hashing O(1): dictionary

What’s trouble of hashing function?

Collision:

If H(k1)=H(k2), there is a trouble

Hash: even(k)=1, 0? Worst all even k will collide;

Optimal hashing function:

Mod prime, reshuffle bits: avoid collisions

Math: best hashing H(k) should be a uniformly distributed as possible

Disinterested monkey randomly throw the balls to different bins

For example:

N bins b1~bN

N balls:

For one bin k (random): to be empty p(bin k is empty): after n random throws of n balls to n bins

Divide & contral:

1. After the 1st throw, p1(k is empty) = 1 – 1/n
2. … 2nd thros, p2 ( k is still empty)=(1-1/n)^2

..

n. After nth pn(…)=(1-1/n)^n

lim x->oo (1-1/x)^2 = 1/e = 0.37 (37 per cent rule)

collide always happen

in hashing: increase the number of bins

O(log log n)

O(1)

Small data: hashing

DB: sorting still alive in DB and big data

In CS: ML/DL Bayesian stats

2 types of stats:

classic stats vs Bayesian (subjective) stats: P(X|Y) conditional

can update out belief based on evidence: ML/DL

everything is objective

9.21 Thursday

P(T|E): posterior probability of a theory T given evidence E

P(T): prior probability (belief)

P(E|T): likelihood probability given theory the likelihood of E

P(E): marginal

P(T|E): P(T,E)/P(E) = P(T)P(E|T)/P(E)

Wet weekend: either Sunday or Saturday or both days

Fear for his life to publish this formula

Weatherman: Saturday: 50%; Sunday: 50%

he claim: for sure we will have a wet weekend

P(wet weekend)=0.75

P(wet weekend) = 1 – P(dray weekend)=1-p(no rain on Saturday)\*P(no rain on Sunday)=1-0.5\*0.5=0.75

P(speeder captured by police) = 0.01 (1 percent)

P(prof thief to be arrested) = 0.01

Today A is not captured, A to be captured tomorrow: 0.01

Independent

P(tomorrow cap|today cap) = P(tomorrow cap)

What is the probability to be captured if A keep speeding/stealing for a whole year.

P(not captured) = 1 – 0.01 =0.99

1-0.99^365=1

law of court: murder: 12 juror: 0.5

0.5^12

on-line recording:

sufficient stats: avg temp so far sumi=1 to n Xi/n

don’t want to save Xi’s one by one in a huge table

why saving all of them (measurement)

avg\_o is sum of x1 up to Xn-1

avg\_n is sum of x1 up to Xn

relation between avg\_o and avg\_n

(n-1)\*avg\_o = x1+x2+…x\_n-1

n\*avg\_n= x1+x2+…x\_n

n\*avg\_n = (n-1)\*avg\_o +Xn

avg\_n=(n-1)avg\_0-avg\_n) +Xn

new avg = (n-1)\*avg\_diff + Xn

Natural language ambiguous: misunderstanding

Software engineering:

Software crisis:

Feasibility study: math /stat modeling, solvable

Turing machine equivalent: O(n\*\*K) polynomial

1.1\*\*n infeasible

NP-hard: enumerate all possible solutions, to verify is polynomial

Non-deterministic polynomial

To verify generally is easy O(n\*\*k)

Find the candidate/suspect is hard

Equivalent to a certain known NP hard: traveling saleman’s problem (TSP)

50 cities to visit: optimal route once and only once

non-deterministic polynomial

A murder crime: find out who did it

“easy” brute-force:

arrest everyone

install cameras everywhere, police work will be easy

moving target: lack of visualization, clients feel entitled to ask dev to change the space/des/sys .

mutual misunderstanding: clients and developer, both are too confident, there is a mismatch of the true meaning

gas station: replace the handle after you are done.

Replace: put it back

Reduce mm as much as possible: huge danger

Approaches:

1. More communication (more nature language, distribution) induce more ambiguity

2. common language to improve understanding

3. universal language without any possible ambiguity: math/logic

equation will nail down the ambiguities

4.power means: diagram/figure: many diagrams in engineering

finite state Machine: FSM regular language

update avg: use space efficient method:

n\*n\_avg = (n-1)o\_avg + Xn

n\_avg = (1-1/n)o\_avg +1/n\* Xn

\eta =1/n

if n is small: if is large

if n is large: if is small 1/n almost zero

n\_avg = o\_avg +(Xn-o\_avg)/n

new\_parameter = old\_parameter + learning \_factor\*diff

discount the experience

new\_belieff = old\_belief +\eta \* diff: Bayesian formula

learning\_factor:

37%: in hashing function

explore & exploit policy:

causal process:

make a decision on the point

job-interview policy

100 candidates, each one interviewed for 1 day

what is the optimal policy to decide the winner (get job offer)

ensure the best (possible) candidate to be hired

100days, call the winner? Have complete info, no success

if you find the current one is great, make offer, regret later, not optimal: exploit too early

best policy: interview 50%, in the latter half, make offer to anyone who is better than the best in the 1st half

explore +exploit : explore 50%; exploit; 2nd half

37% rule: explore 1/e percent

63% exploit proved this is the best (stat)

leave some time to explore 37%

test/mock interview: 37%

software quality measure:

efficiency: low in space + time

user-friendly: easy to use by non-cs floks

understandability: peers/self read src code easily

how to achieve: pseudo- English ; var/func names are meaningful, add comments

each function ; 10 lines one sentence; new ds

python: bragging right: one-liner: hurt understandability

peer obs version:

no indentation:

programming : 1. Use same language

1st gen programming language: machine code (binary) cpu: punch/carry cards

2nd gen: assembly: language: require by all cs dept

benefit of assembly:

010101 add AX,01 improved the understandability

* 1. correspondence between machine code assembly

has its own benefits: relevant; size of assembly very small

3rd gen: structural programming language; math proof:

1. type of stmt to sovle TME problem
2. iteration: loop/recursion
3. conditional: If else
4. sequence: in C/C++

pascal: wirth: turing Dijkstra

pure 3rd gen Programming language: no goto/jump

IT/CS industry: no one use pascal too pure to be useful

Ugly but useful c:2.5 gen:assembly+pascal

Organize program into procedure: better reusability

1000 line function: to better reuse or understand

4th generation:

1. SQL: Database
2. Non TM equivalent for Database(pandas/polars)

Declarative PL: killer of cs/programmers

Write what to do, PL take care the “how” part

Req:print out all boys whose gpa is higher than 3.5

```sql

Select name

From std

Where gender == ‘M’ and gpa >3.5;

```

no general programming language:

prolog: too different

how part is handled by SQL:

meta-data/stats: improve search efficiency

DB: has a bar/threshold: no redundancies

Quiz : 10/5

Data warehouse: mongoDB:

hotness dictionary:

Obama;

1995: search Obama: small town in japan

In ccny: gpa > 3.5 then gender

Each condition is a filter

Most selective filter will get small # of success

9.28.23

4th gen PL

1. SQL: declarative: what/how part: meta-data
2. Object-oriented PL: java, Ada(DoD), C++: 3.5 gen PL /C#:main

Python: muti-paradigm

Torch, init, forward,

Define a class

Most important /killer feature of oo: inheritance: reusability

Live with inheritance: cut & paste

Localized features

There is no correct software system:

No proof

Test

Our system never proved, testing is incomplete induction

Poticially correct way:

Our software system is good to our best knowledge in good faith—small font; if you find bugs. Let us know, we’ll try our best to fix

System buggy by nature, find bug, fix once and for all

Hard to traverse/track your bugs

1+2+3+… +n =n(1+n)/2

can’t proof by example

prove by math(complete) induction:

n=1

assume n =k is right

prove n = k+1

prove by contradiction

Euclid: prove the prime numbers are infinite

Assume the # prime numbers is finite

P1(2), P2(3),P3(5), …,Pn(very)

New\_number = P1\*P2\*…P(n+1)

A new prime > “supposedly” largest on prime number

Contradiction!

Beside inheritance: dynamic blinding

3rd : functional programming: lisp

function programming: golang, scala

function is the 1st class object: higher order function

multi-paradigm:

for any function, there is no side effect: most important feature of function programming

for big data: google/meta/amazon: many servers run many function

no loop: use recursion

Program language: be defensive

Communicate with users: be careful for error/ be nice to users

Communicate with operating system: be careful

Allocate: assume success, == null

Try

Command functional feature

Catch

Print(‘…’) error information

Life cycle models: sequences of actions to be done in system development(general)

Water-fall model (1st, introduced by DoD, copycat of conv engineering)

Investigation <-> planning <->designing <-> implementation; integration <->testing <-> maintenance ->retire/death

Feasibility: deliver/make $$/contribute to the goodness of humanity

Numpy: matlab

Vectorized computing

a=[1,2,3]

a+3;

a=np.arange(3)

a+3

planning: $$: best way to ask/demand $$: personnel: who to work with

hardware/ software ; OS; case tools to use: collaborate: git; gui package to use: flask; torch/tensorflow; hugging face, skorch

design: modularize the system (divide & conquer) data structure, alg/logic

2nd phase report

implementation + integration:

coding + debugging

difference between debugging and test:

debugging is itself testing, still biased , testing : by others or self, could more objective:: subjective

testing group is different from dev group

alpha version: already passed internal testing, ok by testing group

beta version: dynamic concept: free users: paid users(expects, matlab)

init: alph;

finally: deliverd version

10.3.23

water-fall

testing: alpha/beta version

google: always beta, collect information about users

maintenance : (after deliverly)

matlab

customer service:

change system: bugs/complains

retirement/death:

cunyfirst: cobol-based system: text prompt

specification is constitution: legal doc

readable to both clients/developer

proceed to next phase: need a system quality assurance (SQA) team to sign off

1st life cycle model( by DoD)

pros: 1st method to copycat engineering principles (success story of engineer) complete doc

spec: plan, design, src, test report, maintenance, report

documentation wirter: science, science engineer

cons: take too long (time), lack of communications between customers/clients and developer

rapid prototyping: improve communication, quickly develop a small system reflecting the gist (major function) very quickly, using (prototyping programming language) such sql, python ; no worry about robustness … other quality can show our understanding

robustness: should not be esay to crash ( defensive programming)

improve communication : between developer

rom- basic

basic based system: 2 coders, one young guy handled the devlop of the feature part

one old guy handled the simulation ( r. p of IBM os)

fast, major features, not reusable

pros: take shorter, improve communication

cons: more change of moving targets

rapid prototyping: not a run-alone model: waterfall+rapid prototyping

evolutionary model: partition system into different builds (4), perform waterfall over each build, better know each other based on real working system reduce the overall mutual risk

pros: rudece overall risk, mutually; improve communication; can have experience of real ( partial) system

cons: (fatal trouble) hard to integrate, have to have many B&f revision, rewriting

spiral model (DoD, wall street):

WF+RP+complete risk analysis in every phase

Pros: reduce risk significantly

Cons: $$ very expensive, good only for mil/ big finance

Rocket: trick to increase the success rate

For a single rocket: probability to success is 90%

How to inprove the success rate:

shoot 2 rockets: failure rate 0.1 -> success: 1-0.1\*0.1=.99

King of life cycle models:

Agile model: tasked whole system, have standup(casual report daily, weekly), sprint ( weekly,bi-weekly, task report, plan next tasks); (month, quarter) PI-meeting: high level os, more serious demo

Lots of meetings

Pros: lots of communication, make programs, improve quality

Cons: too many meetings /distractions/waste time

Why agile model is most popular?

Authority:Project officer,project managers, they can see which people is good.

Quiz:

At last 3 models and cons pros

1. consept question (4 sub question)
2. math/stat/prob model (1)
3. 2 efficient alg design question (2): no coding, pseudo-code to show your ideas

good problem solving skill in efficiency

10/12/2023

skip list(circular double link)

regression(LR)

complexity later for midterm

cousin of the 2-sum question: hashing

when you interview, first choose is: hashing; binary search then dynamic programming, then greedy alg

A[1] < A[N], how to find i s.t A[i]<A[i+1]

1. possible?
2. How to do it, and prove

Assume such I not exist

A[1]>=A[2], A[2]>=A[3]…. A[N-1]>=A[N]

A[1]>=A[N] contradiction

Binary search:

A[mid]

Compare A[1] vs A[mid]

=: A[mid]<A[n] reduced by half, condition true: proceed to the 2nd half

<: A[1]<A[mid]: half, focus on the first half

>: A[1]>A[mid]: A[mid]<A[n]: 2nd half

binary search: safely discard one half, without losing the loop invariant 1st element < last

complexity of alg based on commanding equation

difference equation: discrete value

differential equation ODE ,PDE

for a problem, based on the alg ideas, formulate the commanding eq:

T(n)=1+T(n-1)=1+1+T(n-2)=…=n+T(0) O(n)

T(n)=1+T(n/2)+1+1+T(n/4)=1+1+…T(1)

Merge sort:

T(n)=2T(n/2)+n

=4T(n/4)+n+n=…

2^x=n

x=log2 n

nlogn

fib:

T(n)=T(n-1)=T(n-2)

How to estimate the order of T(n):

a^n =a^(n-1)+a^(n-2)

a^2 =a+1

1.618 >1

1.618\*\*n: infeasible

10.17.2023

mm: math, diagrams

UML:unified modeling (diagram)

3 types: 1. Formal: purely math/logic: finite state machine (FSM) : few people understand

2.informal: natural language; confusing, ambiguity/redundancy:mm

3. semi-formal: combination of diagram, NL, math, program language: most popular NLU/Processing

NLP: deep learning, attention

Attention based deep nets for NLP/U

compression: entropy(jpg, mpg, mp3)

Shannon’s communication theory: robust communication

reliable channel: 40s 50s

need redundancy

AFRL: 3 room: Newton, Einstein, Shannon

Analyze: view-point based analysis: exhaust all possible perspective

Simplest UML (industry standard): use-case diagram **(using in first report)**

Rec: system

Lhs: ord user rhs: prev user

Small ovals inside rec for reatures

Id users/feature/rel btw u/f

Dev fsm: to recognize even numbers (pattern recognition machine)

Alphabet: each state exhaust all possible tokens in the alphab

Name each state by the most recent scanned token so far

Simplified :

Binary number with terminator $, recognize those numbers whose remainder over 3 is 1

4%3=2 fail ,6%3 = 0, 7%3=1 fail

name each state by the remainder fo the value the fsm saw so far

originally 3k+1

shift to the left: (binary) new value after scanning n\_t:

2\*N + new\_token = 2(3k+2) + new\_token = 6k+4+1%3=1

base \*(5/3/7 \*k+rem) +new\_token

3K+2

item 1 :5, 7, or 3 # of states:

item 2: SPC rem: det which state will go to succ/fail

item 3: number system being used: binary: left shift == multiply by 2

octal: left shift == multiply by 8

recognize a certain pattern:

only binary pattern with 010, or 101 then success

binary streams

10.19.23

FSN: remainders over certain number for certain number system

Binary patterns containing 101 or 010: success if present; fail otherwise

Alphabet: 0,1,$

Need to know the count of certain patterns

FSM: SW officially not a UML

Improved cousin of FSM: state chart /diagram: (class diagram)

Life cycle (may ask again in midterm or final) Quick learner (rust, golang), class diagrams, team player,

State chart: use rectangles (no circles)

1-short

Initial state and base state are generally the same, long bar

Have base state: iterate

Allow overlapping pattern (if)

Vending machine: smacks price 30 cents

Allowable coins: 5, 10, 24, p

Semi-formal: use-case diagram: all perspectives

Formal: state chart: logic

Data: normal forms (NF)

Onion shape: 3NF (codd: 1970: Turing) set theory Is the math foundation of DB, Codd NF

Data structure for college student:

Ssn, name , age, gender, dept, college , department-chair, college-president, original-country, pre

Relational DB (RDB) 1970: King of all DB

View of many DBs: keep all information, achieve space/update low complexity

Codd/Boyce: for data , id the key attributes

3NF: all non-key attribute must be directly determined by the key attributes

dept\_chair is not directly determine by ssn, transitive (2NF)

for any 2 tuples, if they share the same value on LHS, then they must have the same RHS

if a data store allows redundant tuples, not even 1NF

dept\_lead\_DB:

dept, dept\_chair

1. Space saving
2. Update: time

College\_lead-data-dict

College, college\_pre

SSN-> department\_chair true

Not directly determine

SSN->dept

Dept->dept\_chair

Different tuple carry valuable information

Class roster:

SSN, course\_id, grade

999, 322, A

888, 322, C

777, 321, B

ssn,course-id,year,semester ->grade 3NF

key: attributes

ssn,course\_id as a key?

ssn cannot be a key

cid: not key

grade: not key