CS. 6240-02 HW5 Vikas Janardhanan

Design Discussion and Pseudo-Code (30 points total) For each step of the algorithm, briefly explain the main idea for how you partition the problem. Show compact pseudo-code for it. It is up to you to decide how to map the computation steps to MapReduce jobs. For example, you can merge multiple steps into a single job, or you can use multiple jobs to implement a single step. Make sure you discuss and show pseudo-code for both versions of step 3. (20 points) Briefly explain how each matrix and vector is stored (serialized) for versions A and B. Do not just paste in source code. Explain it in plain English, using a carefully chosen example. See the above discussion about dense and sparse matrices for an example how to do this. (5 points) Discuss how you handle dangling nodes. In particular, do you simply replace the zero-columns in $\mathbf{M}$ and then work with the corresponding $\mathbf{M}'$, or do you deal with the dangling nodes separately? Make sure you mention if your solution for dangling nodes introduces an additional MapReduce job during each iteration. (5 points)

The problem's starting point is from the input Bz file

Step1: Create adjacency list representation for the pages.
Step 1 and 2:

1. Create adjacency list representation which is PageName – AdjList(Same pseudocode as previous assignment). The Reduce phase does the following:
   a. Compute dead links
   b. Get local pageName-> Number map
   c. Offset with respect to each worker

Partitioning : This job also partitions the dataset into 1/n for each of the worker. So every worker is approximately responsible for parsing 1/nth of the dataset.

Step 1 Job 1: Preprocess pageName – AdjList
//Input: data from .bz file
//Output: PageName – AdjList(Strings)
// Function takes the raw input line from the bz file parses and emits the corresponding pageName to its adjList representation.

```
Map(key k,inputFileLine)
        // remove pages and outlinks containing '~'
        nodeID = parseLineForKey()
        adjList = parseLineForAdjList()
        emit(nodeID,adjList)

Reduce
        Setup()
                Counter = 0
                HashMap<String,Number> pageNameToNumberMap;
        reduce(node n,[AdjLists])
                // Global counter which is incremented if the page has incoming link or has
outgoing links. This is made use of in other jobs
```

```
        pageNameToNumberMap.put(n,counter++)
        for each adjList in adjLists:
                adjListaggr += adjList
        emit(n,adjListaggr)
cleanup()
        emit(context.mapID,pageNameToNumberMap) //local map
        emit(context.mapID,Counter) // to compute offset
```

2. Now using the offset and the map(PageName to number) from previous job, a global unique PageName to PageNumber map is created and vise versa is stored as well for the final step use. This is a map only job.
   Partitioning: This job partitions the input dataset(pageNumber to pageName local map) into 1/n to each worker. So every worker is responsible for forming 1/n of the global hash map.But every map is able to see complete offset data using distributed cache.
   Input: LocalPageNameToNumber mappings, Offsets
   Output: Matrix M

```
   Setup()
        HashMap<Integer,Long> offset = calculateOffset(getOffsetfromHDFS) //using
   relative offset.
        HashMap<String,Long> globalPageNameToNumMap;
   Map(id,localPageNameToNumberMap)
        For each item in localPageNameToNumberMap:
                globalPageNameToNumMap.put(item.name,item.Number+offset.get(id))
   cleanup()
        emit(globalPageNameToNumMap)
        emit(globalNumToPageNameMap)
```

3. Next step is contstruction of the sparse matrix
   A: Row Partition
   Input : PageName – AdjList
   Output: Matrix M
   Mapper

```
        Setup()
                Map<String,Number> PageNameToNumberMap
        Map(String pageName,[adj1,adj2…])
                For each adj in list:
                        emit(PageNameToNumberMap.get(adj),
(PageNameToNumberMap(pageName),1/list.size)

                if list.size ==0 :
                        emit(pageNameToNumberMap.get(pageName)
```

Reducer:
Reduce(pageNumber,[(PageNumber,Contribution) List]
    Emit(pageNumber,List)

Column Partition:
Input : PageName – AdjList
Output: Matrix M
Setup()
    Map<String,Number> pageNameToNumberMap
Map(String pageName, List[Adj1,Adj2,….])
    For each adj in list:
        List rowContrib += [(pageNameToNumberMap(adj)]
    Emit(pageNameToNumberMap(pageName),rowContrib)
    If list.size == 0:
        Emit(pageNameToNumberMap.get(pageName))

4. Matrix Multiplication:
Input: Matrix M
Output: PageRank R
Row Partition: (Map only)
Partitioning: PageRanks and danglingNode number list is copied to every worker where as 1/nth of the rows of matrix end up on a specific machine.

Setup()
    // load the pageRanks,Dangling node number vector from HDFS or 1/n for first iteration
    // load number of pages and iteration count from the context
    Map<Number,Double> PageRanks
    Double danglingNodeContrib
Map(Number pageName,List[(num,contribution)…])
    contributionAggr=0
    for each (num,contribution) in list:
        contributionAggr += PageRanks.get(num)
    PageRank = 0.15/n + 0.85*(contributionAggr+danglingNodeContrib)
    Emit(pageNameToNumber(pageName),pageRank)

Column Partition:
Mapper :
Input: Col – List[RowContributions],PageRank,DanglingNodes
Map(Number,Contribution)
    Emit(colNo,RowContributions)
    Emit(pageNumber,pageRank)
    Emit(DanglingColNumber,null)

Reducer:
Setup()
HashMap<RowNumber,Contribution> aggregator
localDanglingContrib
reduce(Number,[Contributions …])
pageRank = getPageRankEntryFrom([Contributions])
for each contribution in Contributions:
if ValidContribution:
aggregator[contribution.row]+= Contribution.value*pageRank

cleanup()
emit(localDanglingContrib)
emit(aggregator)

Job2:
Mapper:
Input: RowNumber – Contributions
Map(Number,Contribution)
Emit(RowNumber,Contributions)

Combiner:
Combiner(RowNumber, [Contributions])
sum=0
For each contribution in Contributions:
sum+=contribution
emit(RowNumber,sum)
reduce(Number,[Contributions …])
sum =0
for each contribution in Contributions:
sum+=contribution
pageRank = 0.15/TotalPages + 0.85*(sum+globalDanglingContrib)
emit(Number,pageRank)

Top K algorithm:
Same logic as in HW3 . Pseudocode is given there. PriorityQueue is used to compute local topK at mapper and then global topK is computed at the reducer

Only change is the use of pageNumberToNameMap to compute the pageName from the pageNumber while producing the final output.

CS. 6240-02 HW5 Vikas Janardhanan

Briefly explain how each matrix and vector is stored (serialized) for versions A and B. Do not just paste in source code. Explain it in plain English, using a carefully chosen example. See the above discussion about dense and sparse matrices for an example how to do this. (5 points)

Matrix Representation : using sparse representation which means entry is present if and only if it has contributions.

For Row partition:

RowNumber1 [(ColNo,Contribution) …]
RowNumber2 [(ColNo,Contribution) …]

Eg: 3 – [(1,1/3),(2,2/3)]
Which indicates Row 3 has entry for column 1 and 2 with values 1/3 and 2/3 respectively.

For Column Partition:
ColumnNumber1 [(RowNo,Contribution)…]
ColumnNumber2 [(RowNo,Contribution)…]

Eg: 3 – [(1,1/3),(2,2/3)]
Which indicates Column 3 has entry for rows 1 and 2 with values 1/3 and 2/3 respectively.

Dangling Node:
[ColNo1
ColNo2
…]
Where ColNo corresponds to PageNumber corresponding to the dangling Node.

PageRank:
Represented as PageNumber and its associated rank.
[(PageNumber1,PageRank),
[(PageNumber2,PageRank),
…]

Discuss how you handle dangling nodes. In particular, do you simply replace the zero-columns in **M** and then work with the corresponding **M′**, or do you deal with the dangling nodes separately? Make sure you mention if your solution for dangling nodes introduces an additional MapReduce job during each iteration. (5 points)

Since it was observed that dangling mass contribution of each row of pageRank was as a result of D * R which is constant for each iteration, this value is calculated once per iteration. If its

represented in D, it'll be calculated as |N|(number of pages) which creates lot of traffic and waste of resource. Hence the algorithm used is as below:
R(t) = alpha/|N|*|1| + (1-alpha)[M(R(t-1)) + D*R(t-1) |1|]
Where M is the matrix reprentation of the Contribution
R is matrix consisting of PageRank.
N is total pages
Alpha=0.15
D is a vector that contains dangling node numbers.
D*R is sum(1/n*R(D(i))
|1| is identity matrix of N rows.

Performance Comparison (18 points total) Run your program in Elastic MapReduce (EMR) on the full English Wikipedia data set from 2006, using the following two configurations: • 6 m4.large machines (1 master and 5 workers) • 11 m4.large machines (1 master and 10 workers) Report for both configurations the total execution time, as well as the times for (i) completion of steps 1 and 2, (ii) time for an iteration of step 3, and (iii) time for step 4. Make sure you clarify if your program works with the original input files or with the parsed adjacency lists from a previous assignment. Since there are two versions (A and B) for step 3, you need to report 4 sets of running-time numbers. (4 points)

(i) completion of steps 1 and 2

	Row Partition	Column Partition
6 machines	1915332	1851559
11 machines	994858	1063365

(ii) time for an iteration of step 3,

	Row Partition	Column Partition	AdjList
6 machines	55775	215853	131874
11 machines	38745	145391	92844

(iii) time for step 4

	Row Partition	Column Partition
6 machines	41969	46139
11 machines	43766	45911

(iv) Total Time

	Row Partition	Column Partition
6 machines	2507611	3913767
11 machines	1591484	255038

CS. 6240-02 HW5 Vikas Janardhanan

For each configuration, report the running time of an iteration executed by the adjacency-list based Hadoop implementation in the earlier HW assignment. How do these numbers compare to the adjacency-matrix based version? Briefly discuss if you find the result surprising and explain possible reasons for it. Make sure you back your arguments with concrete data, e.g., from the log files. (10 points)

On observing the different approaches, below can be understood and there is no surprise in the same.
Row Partition approach is the fastest among all and Adjacency approach comes next.
Column Partition gives the least performance.
Row Partition advantage: Data shuffling is minimum here, since it's a map only job. This results in almost twice as better performance compared to column partition.
Disadvantage: PageRank is copied to every worker, eventhough data will always be small when compared to matrix data.
Adjancency list approach: This approach had a smart way of handling dangling nodes even though there was a lot of data shuffling during each iteration. However, compared to column partitioning it was not that bad.
The above observations are backed by the below data from the log files:

Row Partition :
Map input records=2189258
Map output records=18
Input split bytes=2754
Spilled Records=0
Failed Shuffles=0
Merged Map outputs=0
File Input Format Counters
Bytes Read=2160511051
File Output Format Counters
Bytes Written=39409236

Adjacency Method:
Map input records=2292317
Map output records=68943773
Map output bytes=3600531676
Reduce input groups=2292318
Reduce input records=68943773
Reduce output records=2292317
HDFS: Number of bytes read=1433478682
HDFS: Number of bytes written=1433476079

Column Method:

Job1

Map input records=5339727
Map output records=135594567
Map output bytes=3661053309
Map output materialized bytes=629573157
Input split bytes=17043
Combine input records=0
Combine output records=0
Reduce input groups=3150469
Reduce shuffle bytes=629573157
Reduce input records=135594567
Reduce output records=0
Job2
Map input records=9450352
Map output records=9450352
Map output bytes=151205632
Map output materialized bytes=117075313
Input split bytes=1323
Combine input records=9450352
Combine output records=9450352
Reduce input groups=2189258
Reduce shuffle bytes=117075313
Reduce input records=9450352
Reduce output records=2189258

Report the top-100 Wikipedia pages with the highest PageRanks, along with their PageRank values, sorted from highest to lowest, for both the simple and full datasets. Are the results the same as in the adjacency-list based Hadoop implementation? If not, try to find possible explanations. (4 points)

Yes the results are the same across all the versions. The pageRanks for adjacency list approach however is slightly different because of the way dangling nodes are handled.

Row Partiton Local
Column Partition Local
0.006268262079423078:United_States_09d4
0.004752925651663509:Wikimedia_Commons_7b57
0.003882083079158817:Country
0.002677639490059997:England
0.00260685204215949:United_Kingdom_5ad7
0.002602586760897884:Europe
0.00258071821115885:Water
0.002536166709396565:Germany
0.002511053183070935:France
0.00245409469342631O3:Animal
0.00242099563757058:Earth

0.002357188856916993:City
0.002007228437722888:Week
0.0019207074411201216:Asia
0.0018678519236924964:Sunday
0.0018585003126376926:Wiktionary
0.0018407354138432956:Monday
0.0018352297868662677:Money
0.0018225276952226308:Wednesday
0.0018102092804128831:Plant
0.0017780532367200103:Friday
0.0017602592546616985:Computer
0.0017583402971368552:Saturday
0.0017746304355422888:English_language
0.0017357191264350184:Thursday
0.0017232944612568 36:Tuesday
0.0017137759164044491:Italy
0.0017032005827194554:Government
0.0017016985990960308:India
0.0015872815219805 62:Number
0.0015584764800810357:Spain
0.0015143848821489222:Japan
0.0014977586737396132:Canada
0.0014701832794905651:Day
0.0014445754534188446:People
0.0014171231784857507:Human
0.0013745140467073553:Wikimedia_Foundation_83d9
0.0013659399391673295:Australia
0.0013655056396149382:China
0.0013330361169104012:Energy
0.001316526313052371:Food
0.0012932737014161406:Sun
0.0012906864348722251:Science
0.0012758619110732963:Mathematics
0.0012472048845615369:index
0.0012255116379364327:Television
0.0011885976311891565:Capital_(city)
0.0011817627345243865:Russia
0.001163482309613943:State
0.00115710837098 0829:Music
0.0011349190516769 52:Year
0.0011118098670028242:Greece
0.0011084419898385145:Language
0.0011054136172328416:Scotland
0.0010817279846470464:Metal

0.0010724538303110826:Wikipedia
0.0010611310344791195:Greek_language
0.0010565256285193832:2004
0.0010307339000191558:Planet
0.0010254766058030561:Sound
0.00102225484040212:Religion
0.0010202595604546826:London
9.90789701996008E-4:Africa
9.555984391173043E-4:20th_century
9.490638415110944E-4:Law
9.435252509644978E-4:Geography
9.370763607187994E-4:Liquid
9.36744537988964E-4:19th_century
9.246934303337457E-4:World
9.228039937697138E-4:Poland
9.127625824780026E-4:Scientist
9.103721224400061E-4:Society
8.778491911297556E-4:Latin
8.777601463480692E-4:Atom
8.759419632775852E-4:History
8.689008813834933E-4:Sweden
8.681683457472661E-4:War
8.646804112637386E-4:Light
8.581913913912197E-4:Netherlands
8.494797122621058E-4:Culture
8.399317320858481E-4:Building
8.23546819632401E-4:God
8.216698391470311E-4:Turkey
8.163021946764484E-4:Plural
8.133630565583003E-4:Information
8.055473764137295E-4:Centuries
7.928525045745853E-4:Chemical_element
7.90814195320167E-4:Portugal
7.870761352325995E-4:Inhabitant
7.77739895308144E-4:Denmark
7.753600656865806E-4:Capital_city
7.708899281390126E-4:Austria
7.589071127376836E-4:Cyprus
7.57069563135634E-4:Species
7.559989237421285E-4:Ocean
7.551197162145302E-4:Book
7.534113103962841E-4:Disease
7.528696397968457E-4:North_America_e7c4
7.505074858768589E-4:University

7.48162315866106E-4:Biology

Row Partition Local:
0.006268262079423116:United_States_09d4
0.004752925651663508:Wikimedia_Commons_7b57
0.003882083079158806:Country
0.002677639490059996:England
0.0026068520421594905:United_Kingdom_5ad7
0.002602586676089786:Europe
0.0025800718211158816:Water
0.0025361667093965713:Germany
0.0025110053183070922:France
0.0024540946934263107:Animal
0.002420995563757064:Earth
0.0023571888569169997:City
0.0020072284377228686:Week
0.0019207074411201198:Asia
0.0018678519236924708:Sunday
0.0018585003126376913:Wiktionary
0.0018407354138432756:Monday
0.0018352297868662677:Money
0.0018225276952226433:Wednesday
0.0018102092804128797:Plant
0.001778053236720008:Friday
0.0017602592546616977:Computer
0.0017583402971368626:Saturday
0.0017463043554228866:English_language
0.0017357191264350414:Thursday
0.0017232944612568398:Tuesday
0.0017137759164044548:Italy
0.001703200582719454:Government
0.0017016985990960308:India
0.001587281521980562:Number
0.0015584764800810327:Spain
0.0015143848821489265:Japan
0.0014977586737396115:Canada
0.0014701832794905539:Day
0.0014445754534188455:People
0.0014171231784857502:Human
0.0013745140467073536:Wikimedia_Foundation_83d9
0.0013659399391673282:Australia
0.001365505639614939:China
0.0013330361169104025:Energy
0.0013165263130523715:Food

0.0012932737014161 41:Sun
0.0012906864348722251:Science
0.0012758619110732963:Mathematics
0.001247204884561536:index
0.0012255116379364319:Television
0.0011885976311891574:Capital_(city)
0.0011817627345243852:Russia
0.0011634823096139417:State
0.0011571083709808298:Music
0.0011349190516769 53:Year
0.0011118098670028233:Greece
0.0011084419898385145:Language
0.0011054136172328425:Scotland
0.001081727984647046:Metal
0.0010724538303110817:Wikipedia
0.0010611310344791 19:Greek_language
0.0010565256285193858:2004
0.0010307339000191558:Planet
0.0010254766058030548:Sound
0.0010222548404021192:Religion
0.0010202595604546813:London
9.907897019960075E-4:Africa
9.55598439117305E-4:20th_century
9.490638415110931E-4:Law
9.435252509644974E-4:Geography
9.370763607188009E-4:Liquid
9.367445379889627E-4:19th_century
9.24693430333745E-4:World
9.228039937697141E-4:Poland
9.127625824780026E-4:Scientist
9.103721224400061E-4:Society
8.778491911297567E-4:Latin
8.777601463480694E-4:Atom
8.759419632775857E-4:History
8.689008813834936E-4:Sweden
8.681683457472657E-4:War
8.646804112637371E-4:Light
8.581913913912206E-4:Netherlands
8.494797122621054E-4:Culture
8.399317320858479E-4:Building
8.235468196323998E-4:God
8.216698391470311E-4:Turkey
8.163021946764482E-4:Plural
8.133630565582988E-4:Information

8.055473764137282E-4:Centuries
7.928525045745853E-4:Chemical_element
7.908141953201678E-4:Portugal
7.870761352326005E-4:Inhabitant
7.777398953081439E-4:Denmark
7.753600656865799E-4:Capital_city
7.708899281390131E-4:Austria
7.589071127376837E-4:Cyprus
7.570695631356339E-4:Species
7.559989237421287E-4:Ocean
7.551197162145302E-4:Book
7.534113103962832E-4:Disease
7.528696397968455E-4:North_America_e7c4
7.505074858768585E-4:University
7.481623158661058E-4:Biology

Row Matrix AWS

0.002895612340091888:United_States_09d4
0.0025837662658379675:2006
0.0013756641642095 9:United_Kingdom_5ad7
0.0011887559248919997:2005
9.331545921970004E-4:Biography
9.001438659434943E-4:Canada
8.946269909617607E-4:England
8.819707744034018E-4:France
8.27461614123789E-4:2004
7.562886831189352E-4:Germany
7.364585608851405E-4:Australia
7.160204348146393E-4:Geographic_coordinate_system
6.664083680790052E-4:2003
6.491299090528725E-4:India
6.337472702443581E-4:Japan
5.37869633110954E-4:Italy
5.361487027045798E-4:2001
5.288554115331115E-4:2002
5.26108855277083E-4:Internet_Movie_Database_7ea7
5.053386276543921E-4:Europe
5.015343994390624E-4:2000
4.815731814636359E-4:World_War_II_d045
4.677556736351032E-4:London
4.47954124626422E-4:Population_density
4.4584862738716115E-4:Record_label
4.429924746813497E-4:1999

4.3990978261103906E-4:Spain
4.3956933153788195E-4:English_language
4.146181651548841E-4:Race_(United_States_Census)_a07d
4.1364953688157694E-4:Russia
4.0666077968928675E-4:Wiktionary
3.85300509800055E-4:Wikimedia_Commons_7b57
3.827812361069518E-4:1998
3.7512127662490827E-4:Music_genre
3.656526984096323E-4:1997
3.6102616020592895E-4:Scotland
3.6045735875986834E-4:New_York_City_1428
3.438433136333664E-4:Football_(soccer)
3.431247424875296E-4:1996
3.3844560924637496E-4:Television
3.380003935095635E-4:Sweden
3.2696341156546557E-4:Square_mile
3.2627172674296295E-4:Census
3.2299669185829546E-4:1995
3.2165967950409613E-4:California
3.1649468075849765E-4:China
3.113372532364528E-4:Netherlands
3.107236747231034E-4:New_Zealand_2311
3.0848837109060447E-4:1994
2.936718608548916E-4:1991
2.9131985809068174E-4:1993
2.8938672813236873E-4:1990
2.8901262991029494E-4:New_York_3da4
2.8844123039015884E-4:Public_domain
2.7906163398987963E-4:1992
2.7877882733432776E-4:United_States_Census_Bureau_2c85
2.778208315536224E-4:Film
2.759587905166995E-4:Scientific_classification
2.754066917347184E-4:Actor
2.725732034022616E-4:Norway
2.7167995761467027E-4:Ireland
2.649319762740505E-4:Population
2.617771297995301E-4:1989
2.5578979295047424E-4:1980
2.555902889562312E-4:Marriage
2.5503294304727876E-4:January_1
2.5429178563541493E-4:Brazil
2.529464259858673E-4:Mexico
2.519142612574905E-4:Latin
2.4902485950670374E-4:1986

2.469965868943201E-4:Politician
2.4277841622043054E-4:1985
2.4240004633174956E-4:1979
2.418948989627121E-4:1982
2.4173203771540546E-4:French_language
2.4157035425746552E-4:1981
2.4117545225030312E-4:Per_capita_income
2.398090623359567E-4:Poland
2.3948752284683568E-4:1974
2.3937069933898696E-4:Album
2.3779856609789526E-4:South_Africa_1287
2.3722845366096158E-4:Switzerland
2.3714836890721179E-4:1984
2.371475854077158E-4:1987
2.3698196286943242E-4:1983
2.3576252382891605E-4:Record_producer
2.331483660258783E-4:1970
2.3176630161779766E-4:1988
2.3040742074454973E-4:1976
2.2916979771164594E-4:Km²
2.2749905467755755E-4:1975
2.2478568148506182E-4:1969
2.2454623105836697E-4:Paris
2.2348383281810928E-4:Greece
2.2324046518990042E-4:1972
2.22457357515035E-4:Personal_name
2.2197817108490718E-4:1945
2.213512750751784E-4:1977
2.2040350997510732E-4:Poverty_line
2.203474255931036E-4:1978

Col Matrix Partition AWS
0.0028956123401003763:United_States_09d4
0.00258376626584564:2006
0.0013756641642137626:United_Kingdom_5ad7
0.0011887559248954594:2005
9.331545921998567E-4:Biography
9.001438659462171E-4:Canada
8.946269909644885E-4:England
8.819707744059608E-4:France
8.27461614126194E-4:2004
7.562886831211407E-4:Germany
7.364585608873847E-4:Australia

7.160204348168972E-4:Geographic_coordinate_system
6.664083680809288E-4:2003
6.491299090547553E-4:India
6.337472702462888E-4:Japan
5.378696331124942E-4:Italy
5.36148702705919E-4:2001
5.288554115345774E-4:2002
5.261088552787581E-4:Internet_Movie_Database_7ea7
5.053386276558755E-4:Europe
5.015343994406154E-4:2000
4.8157318146504064E-4:World_War_II_d045
4.6775567363649574E-4:London
4.479541246278205E-4:Population_density
4.4584862738865095E-4:Record_label
4.429924746825135E-4:1999
4.39909782612269E-4:Spain
4.3956933153913307E-4:English_language
4.146181651561182E-4:Race_(United_States_Census)_a07d
4.136495368827698E-4:Russia
4.066607796904505E-4:Wiktionary
3.8530050980115343E-4:Wikimedia_Commons_7b57
3.8278123610804767E-4:1998
3.7512127662612117E-4:Music_genre
3.6565269841070395E-4:1997
3.610261602070111E-4:Scotland
3.6045735876094175E-4:New_York_City_1428
3.438433136344362E-4:Football_(soccer)
3.431247424885489E-4:1996
3.384456092474221E-4:Television
3.380003935105678E-4:Sweden
3.269634115664916E-4:Square_mile
3.2627172674400725E-4:Census
3.2299669185922386E-4:1995
3.2165967950507094E-4:California
3.1649468075940827E-4:China
3.1133725323736436E-4:Netherlands
3.107236747240407E-4:New_Zealand_2311
3.084883710915099E-4:1994
2.936718608557361E-4:1991
2.9131985809153376E-4:1993
2.89386728133221E-4:1990
2.8901262991116804E-4:New_York_3da4
2.8844123039101373E-4:Public_domain
2.790616339906926E-4:1992
2.7877882733520103E-4:United_States_Census_Bureau_2c85
2.778208315545003E-4:Film

2.759587905175728E-4:Scientific_classification
2.7540669173558115E-4:Actor
2.725732034030807E-4:Norway
2.716799576154858E-4:Ireland
2.6493197627485077E-4:Population
2.6177712980029756E-4:1989
2.5578979295120136E-4:1980
2.5559028895704556E-4:Marriage
2.5503294304797877E-4:January_1
2.542917856361653E-4:Brazil
2.5294642598662346E-4:Mexico
2.519142612581927E-4:Latin
2.49024859507425E-4:1986
2.4699658689508817E-4:Politician
2.4277841622114015E-4:1985
2.424000463324337E-4:1979
2.4189489896340948E-4:1982
2.4173203771607517E-4:French_language
2.4157035425815849E-4:1981
2.4117545225108242E-4:Per_capita_income
2.39809062336658E-4:Poland
2.3948752284751383E-4:1974
2.3937069933978764E-4:Album
2.3779856609859324E-4:South_Africa_1287
2.3722845366166306E-4:Switzerland
2.371483689079057E-4:1984
2.3714758540840946E-4:1987
2.3698196287010614E-4:1983
2.357625238296882E-4:Record_producer
2.3314836602653593E-4:1970
2.317663016184707E-4:1988
2.3040742074520727E-4:1976
2.2916979771237423E-4:Km²
2.274990546782017E-4:1975
2.2478568148569174E-4:1969
2.2454623105901134E-4:Paris
2.234838328187604E-4:Greece
2.2324046519052885E-4:1972
2.2245735751573234E-4:Personal_name
2.2197817108552135E-4:1945
2.2135127507581459E-4:1977
2.2040350997581305E-4:Poverty_line
2.2034742559373183E-4:1978

CS. 6240-02 HW5 Vikas Janardhanan