## Design Discussion and Pseudo-Code

***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)***

**Step 1 : Create matrix $\mathbb{M}'$ from either the original input or the adjacency-list representation from a previous assignment. (But also see the discussion about sparse matrix representation below!)**

We create the adjacency lists as we have done before in previous assignments. i.e. PageName: Adjacency List. The reduce call performs following functions

a) Compute dead links
b) Offset w.r.t to each worker
c) Getting local pageName : Number map.

The partitioning ensures that each worker gets 1/nth of dataset

The pseudo code for Step 1 can be visualized as follows

```
Map (key k, line l){
        nodeId = getKey(l) // Parser Given as part of homework
        adjList = getAdjacencyList(l)
        emit(nodeId, adjList)
}
```

```
Class Reduce{

      Method Setup(){
      Counter = 0;
      HashMap<String, Number> nameToNumMap;
      }

      method reduce(node_name n, List[adjLists] ){
                StringToNumMap.put(n, Counter++)
                for each adjList in adjLists
                aggrAdjLinks += adjList
                emit(n, aggrAdjLinks)
          }

   method Cleanup(){
       emit(MapId, StringToNumMap
       emit(MapId, Counter)
   }
} //end of reduce class
```

Now, using the offset and local map, we create a unique global map between pageName and Number. This is easily done using a map only job. As before , the job is partitioned such that each worker gets to compute global map for 1/nth of inputs. However every map sees complete offset data using distributed cache. The pseudo code for the above process is as follows:

```
Class Mapper {

        HashMap globalMap ;
        HashMap offset


        Method Setup(){
            HashMap offset = getOffset(pathToHdfs);
        }

        Method map(id, localMap){
                For each element e in localMap:
                        globalMap.put(e.name,e.number+offset(id))
        }

        Method cleanup(){
           Emit(globalMap);
           Emit(inverted(globalMap));
        }

}
```

After this step, we have all the files needed to construct sparse matrix . We do this in two ways

Method 1:

Column Partition:

To form a column we just need adjacency list and hence it is a map only job. As before, we partition the data so that each worker gets 1/nth of data. The pseudo code is as follows

```
Class Map{

        Method setup(){
                Map<String, Number> nameToNumberMap = load Map from
                hdfs
        }

        method map(String pageName, List[Adj1, Adj2 .. ]){
                for( each adj in list){
                        List rowContribution += [(nameToNumberMap (adj)]
                }

        emit(nameToNumberMap (pageName), rowContribution);

        if(list.size == 0){
                emit(nameToNumberMap.get(pageName))
        }
}
```

Method 2 : Row Partition

```
Class Map{
        setup(){
                Map NameToNumber = load from hdfs
        }

        method map(String pageName, List[Adj1, Adj2 .. ]){
                for( each adj in list){
                        emit(NameToNumber.get(adj),
                        (NameToNumber(pageName), 1/list.size))
                }
                if(list.size == 0){ // write to dangling vector on HDFS
                        emit(NameToNumber.get(pageName))
                }
        }

        cleanup(){
           // Do Nothing
        }
}

method Reduce(PageNumber, [(PageNumber, Contributions),List]){
        Emit(PageNumber, List)
}
```

**Step 2: Create vector $\mathbf{R}(0)$ with the initial PageRank values $1/|V|$.**

Created automatically during 0th iteration.

**Step 3: Run 10 iterations to compute $\mathbf{R}(10)$.**

Method 1 : Column Partition
We use two jobs to perform the actions

Job1 :

```
Method Map(Number, Contribution){
        emit(ColNumber, RowContributions)
        emit(PageNumber, PageRank)
        emit(DanglingColNumber, null)
}

class Reducer{
        HashMap aggregator;
        LocalDanglingMass;

        Method setup(){
                //do nothing
        }

        method reduce(Number, [Contributions…]){
                double pageRank = get PageRankEntryFrom[Contributions];
                for (each Contribution in Contributions){
                        if(ValidContribution){
                                aggregator[Contribution.row] += Contribution.value*
                        pageRank;
                        }
        }
```

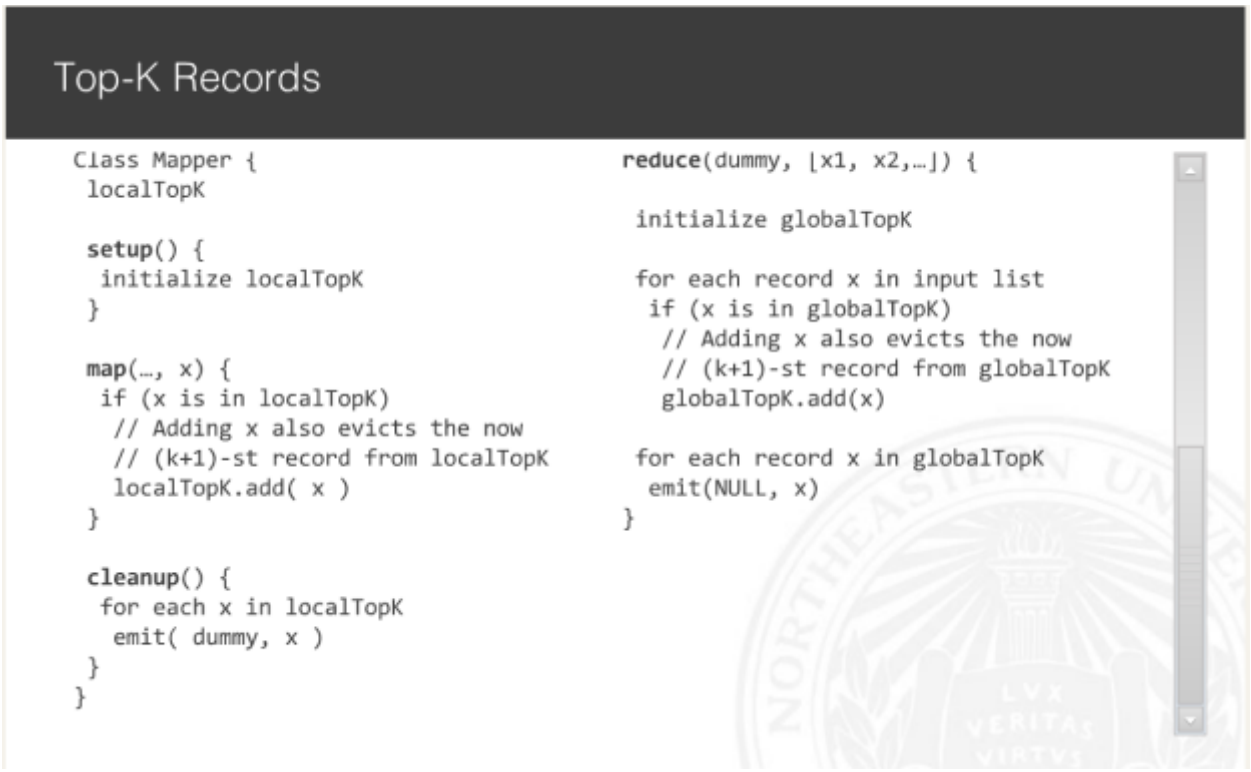
```
        CleanUp(){
                Emit(localDanglingMass);
                Emit(aggregator);
        }

Job 2:

Method Map(Number, Contribution){
        Emit(RowNumber, Contributions)
}

method Combiner(RowNumber,List [Contributions]){
        Sum ← 0
        For each Contribution c in List {
                Sum ← sum + c
        }
        emit(RowNumber, Sum)
}

class Reducer{
        method Setup(){
                GlobalDanglingMass ← input from output of job 1
}
reduce(Number, List [Contributions…]){
        Sum ← 0
        For  each Contribution  c in List :
                Sum + = Contribution
        PageRank = 0.15/NumberOfPages + 0.85 *(sum + GlobalDanglingMass)
        Emit(Number, PageRank)
}
```

Method 2: Row Partition . this method is a map only job

```
Class Map{
        Map PageRanks
        Double danglingMass = computeDanglingMass

         setup(){
                //do nothing
        }

        map(Number pageNum, List[(num, contribution) .. ]){
                contributionAggr = 0
                for( each (num, contribution) in list){
                        contributionAggr += PageRanks.get(num) *
                }
                PageRank = 0.15/n + 0.85 (contributionAggr + danglingMass )
                emit(PageNameToNumber(pageName), rowContribution);
        }

        cleanup(){
          //do nothing
        }
}
```

Step 4: Return the top-100 pages from $\mathbf{R}(10)$.

This step is implemented using Top-K design pattern and uses same approach mentioned in modules. The pseudo code from modules is as follows:



Top-K Records

```
Class Mapper {
 localTopK

 setup() {
  initialize localTopK
 }

 map(…, x) {
  if (x is in localTopK)
   // Adding x also evicts the now
   // (k+1)-st record from localTopK
   localTopK.add( x )
 }

 cleanup() {
  for each x in localTopK
   emit( dummy, x )
 }
}
```

```
reduce(dummy, [x1, x2,…]) {

 initialize globalTopK

 for each record x in input list
  if (x is in globalTopK)
   // Adding x also evicts the now
   // (k+1)-st record from globalTopK
   globalTopK.add(x)

 for each record x in globalTopK
  emit(NULL, x)
}
```

***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)***

The serialization for column partition is as follows

ColumnNumber1 [(RowNumberI,Contribution)…….]
ColumnNumber2 [(RowNumberJ,Contribution)…….]
.
.
.
ColumnNumberN[(RowNumberA,Contribution)…….]

Example
10[(6,0.4),(4,0.1),(7,0.7)]

The serialization for row partition is as follows

RowNumber1 [(ColumnNumberI,Contribution)…….]
RowNumber2 [(ColumnNumberJ,Contribution)…….]
.
.
.
RowNumberN[(ColumnNumberA,Contribution)…….]

Example
10[(6,0.4),(4,0.1),(7,0.7)]

***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)***

The dangling mass contribution for each row is equal to D*R which Is a constant . Hence we calculate it once per iteration.

## Performance Comparison

***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)***

The solution works with original input files.

The running times observed as follows

Step 1 and Step2 (Combined):

Column Partition :

6 Machines : 1861285ms
11 Machines: 1073273ms

Row Partition :

6 Machines: 1925322ms
11 machines: 997864ms

Step 3:

Column Partition:

6 Machines: 223412ms
11 Machines: 151263ms

Row Partition:

6 Machines: 56874ms
11 machines: 39062ms

Adjacency List:

6 Machines: 129873ms

11 machines: 94558ms

Step 4:

Column Partition:

6 Machines: 46128ms

11 Machines: 45826ms

Row Partition:

6 Machines: 42564ms

11 machines: 43756ms

Total Time:

Column Partition:

6 Machines: 3924162ms

11 Machines: 2602148ms

Row Partition:

6 Machines: 2506681ms

11 machines: 1611825ms

***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)***

The results are in-line with our expectation . In genral , row based partition performs fastest whereas adjacency-list based implementation is second . The column based approach is the worst performing.

The data logs for the approaches are as follows

## 1) Row Partition approach:

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

## 2) Adjacency Based Approach:

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

## 3) Column Partition 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

***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)***

The results are same as the results from assignment 4. The top 100 pages are as follows.

Local Column Implementation :

0.006270017474887648 United_States_09d4
0.004746521647459634 Wikimedia_Commons_7b57
0.0038798607271342137 Country
0.00268094437760841 England

0.002607338111311537 United_Kingdom_5ad7
0.0025985862902878553 Europe
0.002583908182959152 Water
0.0025349693900029984 Germany
0.0025115882823210215 France
0.0024576731944577685 Animal
0.002423856703657666 Earth
0.0023498665911180923 City
0.002007600370654787 Week
0.0019192207972314231 Asia
0.0018681364480376227 Sunday
0.0018619013160561624 Wiktionary
0.0018410203611934517 Monday
0.00183596863517039 Money
0.0018228086846631287 Wednesday
0.0018141869460735083 Plant
0.0017783378690091893 Friday
0.00176330567132573 4 Computer
0.0017586117009649768 Saturday
0.0017474216199944424 English_language
0.0017359752571455965 Thursday
0.0017235388050616633 Tuesday
0.0017146957999186463 Italy
0.0017018287899640493 Government
0.001701025547124364 8 India
0.0015895381781201273 Number
0.0015576674302577678 Spain
0.001515455854610598 Japan
0.0014979884203456966 Canada
0.0014706001107316965 Day
0.0014454183366565565 People
0.0014190431134654758 Human
0.00137369955481059 Wikimedia_Foundation_83d9
0.0013667870686872418 Australia
0.001365576351834140 7 China
0.0013354508748833055 Energy
0.001319343212293935 Food
0.0012950569993510348 Sun
0.0012918919151513764 Science

0.0012775185841551401 Mathematics
0.0012282471356595296 index
0.0012273536593279245 Television
0.0011882683246315577 Capital_(city)
0.0011808000010392096 Russia
0.001163827941607846 State
0.0011570661576948665 Music
0.0011358675592467798 Year
0.0011122234112611738 Greece
0.001106701576836453 7 Scotland
0.0011041364469016755 Language
0.00108401687810087 Metal
0.0010701037513338616 Wikipedia
0.0010622717369057184 Greek_language
0.0010512801859716356 2004
0.0010320329748448626 Planet
0.0010256749088513194 Sound
0.0010237591961046416 Religion
0.0010208486668707758 London
9.91968669115096E-4 Africa
9.544869629369787E-4 20th_century
9.500184614578627E-4 Law
9.430356425999918E-4 Geography
9.388921587115989E-4 Liquid
9.365804297985676E-4 19th_century
9.251877891523698E-4 World
9.143875732073568E-4 Scientist
9.108073281798581E-4 Society
8.797103674045314E-4 Atom
8.789441517334165E-4 Latin
8.756352011229699E-4 History
8.685252408615898E-4 War
8.684172930340793E-4 Sweden
8.681213509695934E-4 Poland
8.662307127425786E-4 Light
8.582735823099307E-4 Netherlands
8.488162924455452E-4 Culture
8.41160157213468E-4 Building
8.247629795706092E-4 God

8.192086777479027E-4 Turkey
8.169803862799127E-4 Plural
8.125725939261808E-4 Information
8.057041813788021E-4 Centuries
7.947276814964438E-4 Chemical_element
7.906159051543335E-4 Portugal
7.809073542051404E-4 Inhabitant
7.771310938113481E-4 Denmark
7.73586365724581E-4 Capital_city
7.703195785370811E-4 Austria
7.586412399089749E-4 Species
7.578020413576397E-4 Cyprus
7.566499018373528E-4 Ocean
7.550868120552979E-4 Book
7.550238738104453E-4 Disease
7.536968663668934E-4 North_America_e7c4
7.495640356183109E-4 Biology
7.49551416482597E-4 University

Local Row Implementation:

0.006270017474887574 United_States_09d4
0.004746521647459629 Wikimedia_Commons_7b57
0.003879860727134216 Country
0.0026809443776084126 England
0.0026073381113115393 United_Kingdom_5ad7
0.0025985862902878584 Europe
0.002583908182959156 Water
0.0025349693900029984 Germany
0.0025115882823210393 France
0.002457673194457764 Animal
0.002423856703657672 Earth
0.0023498665911180962 City
0.0020076003706547673 Week
0.0019192207972314262 Asia
0.0018681364480375954 Sunday
0.001861901316056162 Wiktionary
0.0018410203611934376 Monday
0.0018359686351703906 Money

0.0018228086846630 94 Wednesday
0.0018141869460735 07 Plant
0.0017783378690091481 Friday
0.0017633056713257354 Computer
0.0017586117009649052 Saturday
0.0017474216199944413 English_language
0.0017359752571455788 Thursday
0.0017235388050616002 Tuesday
0.0017146957999186524 Italy
0.0017018287899640502 Government
0.0017010255471243644 India
0.0015895381781201264 Number
0.0015576674302577693 Spain
0.0015154558546106004 Japan
0.0014979884203457005 Canada
0.0014706001107316843 Day
0.0014454183366565591 People
0.0014190431134654756 Human
0.0013736995548105902 Wikimedia_Foundation_83d9
0.0013667870686872427 Australia
0.0013655763518341383 China
0.0013354508748833038 Energy
0.0013193432122939341 Food
0.0012950569993510342 Sun
0.0012918919151513753 Science
0.0012775185841551403 Mathematics
0.0012282471356595299 index
0.0012273536593279241 Television
0.0011882683246315603 Capital_(city)
0.0011808000010392 06 Russia
0.0011638279416078 45 State
0.0011570661576948671 Music
0.0011358675592467794 Year
0.0011122234112611743 Greece
0.0011067015768364543 Scotland
0.0011041364469016738 Language
0.0010840168781008702 Metal
0.0010701037513338616 Wikipedia
0.0010622717369057 19 Greek_language

0.0010512801859716356 2004
0.0010320329748448635 Planet
0.0010256749088513196 Sound
0.0010237591961046399 Religion
0.0010208486668707754 London
9.919686691150947E-4 Africa
9.544869629369787E-4 20th_century
9.500184614578623E-4 Law
9.430356425999924E-4 Geography
9.388921587115984E-4 Liquid
9.365804297985694E-4 19th_century
9.251877891523699E-4 World
9.143875732073565E-4 Scientist
9.108073281798569E-4 Society
8.797103674045316E-4 Atom
8.789441517334165E-4 Latin
8.75635201122971E-4 History
8.685252408615899E-4 War
8.684172930340793E-4 Sweden
8.68121350969594E-4 Poland
8.662307127425791E-4 Light
8.582735823099305E-4 Netherlands
8.48816292445545E-4 Culture
8.411601572134674E-4 Building
8.24762979570608E-4 God
8.19208677747904E-4 Turkey
8.169803862799111E-4 Plural
8.125725939261813E-4 Information
8.057041813788026E-4 Centuries
7.947276814964444E-4 Chemical_element
7.906159051543348E-4 Portugal
7.809073542051401E-4 Inhabitant
7.771310938113499E-4 Denmark
7.735863657245808E-4 Capital_city
7.703195785370825E-4 Austria
7.586412399089746E-4 Species
7.578020413576398E-4 Cyprus
7.566499018373539E-4 Ocean

7.550868120552983E-4 Book
7.550238738104447E-4 Disease
7.536968663668928E-4 North_America_e7c4
7.495640356183104E-4 Biology
7.495514164825975E-4 University

EMR Column Partition Output :

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

EMR Row Output :
0.002895612340091888 United_States_09d4
0.0025837662658379675 2006
0.00137566416420959 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