# Homework3

Name: Yang Cai
NUID: 001632759

## Design Discussion

### Pseudo code of Preprocessing

```
map(line) {
        // this is the same as the parser in the example provided
        pageName, html = parse(line)
        // there will be some parsing errors in parsing if we don't do this
        html = html.replace("&", "&amp;");
        // this is the same as the parser provided by professor
        List<String> linkPageNames = xmlReader.parse(line)
        emit(pageName, linkPageNames)
        for (String name : linkPageNames) {
                // for pages that will only appear in links, but page itself does not exist in data
                // emit empty adjacent list
                emit(name, new List<>())
        }
}
reduce(page, List< linkPageNames> listOfAdjacentList) {
        // increment global counters to get the total number of nodes
        globalCounters.get(totalNodes).increment(1)
        List<String> adjList = new List<>();
        for (list : listOfAdjacentList) {
                // merge all adjacent lists pointed by one page
                adjList.addAll(list)
        }
        // remove all self-link
        while(adjList.remove(key))
        emit(key, adjList)
}
```

### Pseudo code of PageRank

```
map(key, value) {
        // reconstruct Node object from text string
        Node n = Node.fromString(value)
        // these two are set in main function, totalNodes is total number of pages,
        // firstIteration  is a Boolean representing whether it is first iteration
        totalNodes = config.getLong(totalNodes)
        firstIteration  = config.getBoolean(firstIteration )
        // if it is first iteration, then set page rank as initial value 1/totalNodes
        // otherwise read the page rank from input file
        pageRank = firstIteration  ? 1.0 / totalNodes : n.getRank()
        emit(key, n)  //  emit node itself
```

```
        if (n.adjacencyList > 0) {  // if it is not dangling node
                p = pageRank / n. adjacencyList.size
                for (linkPagename : n.adjacencyList) {
                        emit(linkPagename, p)
                }
        } else {
                // We are using order-inversion here: for each dangling node, we emit one
                // dummy node for each reducer task. In the Reducer, it will first sum up all
                //page ranks of dummy node, which equals sum of dangling nodes' pagerank
                for i = [0….config.numOfPartition] {
                        emit(dummy_i, pageRank)
                }
        }
}

Class Reducer {
        delta = 0.0 //sum of dangling nodes' page rank

        reduce(page, List<Node> values) {
                alpha = config.getDouble(alpha), totalNodes = config.getLong(totalNodes)
                sum = 0
                // if it is dummy node, then sum up all the page ranks, which should be
                // dangling nodes' page ranks
                if (isDummyNode(page)) {
                        for (v : values) delta += v
                        return
                }
                // if it is not a dangling node, then we can calculate its pagerank
                Node node = null
                for (v : values) {
                        if (isNode) node = v
                        else sum += v
                }
                sum += delta / totalNodes
                n.pagerank = alpha / totalNodes + (1 – alpha) * sum
                emit(page, n)
        }
}

Class KeyComparator {
        int compare(n1, n2) {
                int res = n1.compare(n2)
                // we have to ensure dummy node comes before any other nodes in Reducer
                if (res != 0 and isDummyNode(n1)) return -1
                if (res != 0 and isDummyNode(n2)) return 1
                return res
```

```
        }
}

Class KeyPartitioner {
        int getPartition(k, v) {
                // we have to ensure each Reduce task receive exactly one dummy node
                if (isDummyNode(k)) return getDummyIndex(k)
                return k.hash % partition
        }
}
```

## Pseudo code of TopK

```
Class Map {
        PriorityQueue<Node> pq;

        map(key, value) {
                // reconstruct Node object from text string
                Node n = Node.fromString(value)
                pq.add((key,n))
                // use PriorityQueue to maintain top 100 pages in each mapper
                // pop out the 101st if there are 101 elements already
                if (pq.size > 100) pq.pollSmallest()
        }

        cleaup() {
                for (tuple : PriorityQueue) {
                        // emit all top 100 pages, using the same dummy nodes
                        // so that they will go to the same reducer
                        emit(dummy, tuple)
                }
        }
}

Class Reducer{
        PriorityQueue<Node> pq;

        reduce(key, List<Tuple> values) {
                for (v : values)        {
                        pq.add(v))
                        // use PriorityQueue to maintain top 100 pages
                        // pop out the 101st if there are 101 elements already
                        if (pq.size > 100) pq.pollSmallest()
                }
        }

        cleaup() {
```

```
                Stack<Tuple> stk
                // push them in stack, so that we can emit them from highest to lowest
                // we can't directly emit them from PriorityQueue, since it is a min heap
                for (tuple : PriorityQueue) {
                        stk.push(tuple)
                }
                index = 0
                // now we can emit them in decreasing order, along with their index
                while (stk is not empty) {
                        emit(index++, stk.pop())
                }
        }
}
```

### Data Transferred

Iteration 1: mapper -> reducer: 13932522, reducer -> HDFS: 5170428
Iteration 2: mapper -> reducer: 13932522, reducer -> HDFS: 5180548
Iteration 3: mapper -> reducer: 13932522, reducer -> HDFS: 5180831
Iteration 4: mapper -> reducer: 13932522, reducer -> HDFS: 5176256
Iteration 5: mapper -> reducer: 13932522, reducer -> HDFS: 5181285
Iteration 6: mapper -> reducer: 13932522, reducer -> HDFS: 5170428
Iteration 7: mapper -> reducer: 13932522, reducer -> HDFS: 5181212
Iteration 8: mapper -> reducer: 13932522, reducer -> HDFS: 5181193
Iteration 9: mapper -> reducer: 13932522, reducer -> HDFS: 5181124
Iteration 10: mapper -> reducer: 13932522, reducer -> HDFS: 5181190

The data transferred from mapper to reducer is exactly the same. The data transferred from reducer to HDFS is almost the same, the reason they are slightly different is that, I write the output as human-readable string instead of binary data, so the result is slightly different due to the double numbers.

## Performance Comparison

### 6 m4.large machines (1 master and 5 workers)

pre-processing time: 34min 37s
10-pagerank time in total: 37min 21s
top-100 time: 71s

### 11 m4.large machines (1 master and 10 workers)

pre-processing time: 20min 42s
10-pagerank time in total: 24min 37s
top-100 time: 40s

### Evaluate the runtime results

1. Preprocessing shows the best speedup.

2. Top-100 phrase shows poor speedup, this is reasonable. Since the data processed is small in this phrase, we filtered out most of the data in the mapper. Also, there is actually only one reducer running, this is because I emit the same dummy node from mappers, so that we can collect all data within one reducer and pick overall top 100 pages. So, this phrase has poor speedup.
   10-pagerank also shows bad speedup, this is might because we are using order inversion, which has really heavily network traffic. Also, we have to compute all dangling nodes' sum on every reducer, which also takes long time.

## Top 100 with simple data size:

The highest pagerank is United_States; 0. 005874025033072947.

0 United_States_09d4: 0.005874025033072947
1 Wikimedia_Commons_7b57: 0.004355334715137075
2 Country: 0.0036385335322493585
3 England: 0.0027015160681119494
4 Germany: 0.0025770984199689397
5 France: 0.0025094140295594137
6 United_Kingdom_5ad7: 0.0025065706185759653
7 Europe: 0.002474293585479647
8 Animal: 0.0023976646464431827
9 Water: 0.0023902803270684577
10 City: 0.0023408647692796467
11 Earth: 0.0021993005344773755
12 Asia: 0.0017731516118374136
13 Wiktionary: 0.0016950250419985344
14 Week: 0.0016924538774798765
15 Money: 0.0016832153661304843
16 Italy: 0.0016725996109017134
17 English_language: 0.0016586887014429354
18 Sunday: 0.0016489805309698874
19 Computer: 0.001634981654795068
20 India: 0.0016305293492473033
21 Monday: 0.0016278662790581053
22 Wednesday: 0.0016125590297969025
23 Government: 0.001579376678442356
24 Friday: 0.0015750879919199535
25 Saturday: 0.0015577051695313023
26 Thursday: 0.0015381618128392585
27 Spain: 0.001529933750546075
28 Tuesday: 0.0015264973725043966
29 Japan: 0.0015150758657973215
30 Plant: 0.0014994413947643542
31 Canada: 0.001441138425001855
32 People: 0.0013964013093158223
33 China: 0.0013498972893048907

Human: 0.001318692959757988
Australia: 0.0013151296483042893
Food: 0.0012997794880418852
Day: 0.0012491542807888147
Number: 0.0011942678226007013
Television: 0.0011929359032842863
Sun: 0.0011927137152275922
Capital_(city): 0.0011601875975458397
Russia: 0.0011594202329779444
Wikimedia_Foundation_83d9: 0.0011496970842856545
Science: 0.0011335241720153895
Mathematics: 0.0011133838875265695
Greece: 0.0011098026259714023
State: 0.0010998688381982434
Year: 0.0010817824650098382
Scotland: 0.0010759696744217341
Metal: 0.0010684128039551656
2004: 0.0010538866993665505
Wikipedia: 0.0010487198787919169
Music: 0.0010323302795617626
Religion: 0.0010133858692189233
Language: 0.0010102337591411136
Sound: 0.0010067523892969073
London: 9.630872547991782E-4
19th_century: 9.235352466504072E-4
Greek_language: 9.138995300639304E-4
Africa: 9.123352965195158E-4
20th_century: 9.106078102673179E-4
Energy: 8.979874928247271E-4
Latin: 8.93410593675727E-4
Planet: 8.901852281828819E-4
Inhabitant: 8.896924185874929E-4
Poland: 8.878870255245126E-4
World: 8.841030442303542E-4
Law: 8.781350890902659E-4
Sweden: 8.571464428853849E-4
Netherlands: 8.500228326232886E-4
Liquid: 8.466453948579979E-4
Society: 8.452219605286949E-4
History: 8.256073975636331E-4
God: 8.171561309846257E-4
Atom: 8.111670663720752E-4
Culture: 8.014401189969187E-4
Scientist: 8.012256873605386E-4
Centuries: 7.997558116035229E-4
Light: 7.94377871373768E-4

80 Capital_city: 7.886004025095171E-4
81 Information: 7.84982108215851E-4
82 Geography: 7.782779207953399E-4
83 Turkey: 7.754757450861012E-4
84 Portugal: 7.727492109723362E-4
85 Plural: 7.674030398647729E-4
86 Austria: 7.52310486215748E-4
87 Book: 7.418333652904868E-4
88 Building: 7.417220290673442E-4
89 North_America_e7c4: 7.38320917207011E-4
90 Chemical_element: 7.341107664926559E-4
91 Electricity: 7.237901917213649E-4
92 U.S._state_5a68: 7.219373965946323E-4
93 Denmark: 7.207241446690024E-4
94 Car: 7.174942088928681E-4
95 Biology: 7.172489056433695E-4
96 2006: 7.14866304060244E-4
97 Ocean: 7.147416375534285E-4
98 List_of_decades: 7.067111596422126E-4
99 Species: 6.988151239491667E-4

## Top 100 with full data size:

The highest pagerank is United_States; 0. 002905009203000658.

0 United_States_09d4: 0.002905009203000658
1 2006: 0.0024878035568702823
2 United_Kingdom_5ad7: 0.001339676371838149
3 2005: 0.001157705891663347
4 Biography: 9.476250144341307E-4
5 Canada: 9.057289404338099E-4
6 England: 8.944846392874961E-4
7 France: 8.608656572412869E-4
8 2004: 8.101758008362058E-4
9 Germany: 7.483097039613323E-4
10 Australia: 7.284702015853869E-4
11 Geographic_coordinate_system: 6.793876298030427E-4
12 2003: 6.524911562223264E-4
13 India: 6.443096014589043E-4
14 Japan: 6.393040461529851E-4
15 2001: 5.241029082769106E-4
16 2002: 5.183937130385478E-4
17 Italy: 5.078321267379116E-4
18 Europe: 4.900772800442649E-4
19 2000: 4.892687119841785E-4
20 Internet_Movie_Database_7ea7: 4.7484746749287175E-4
21 World_War_II_d045: 4.731677424255334E-4

London: 4.484667150345417E-4
1999: 4.325366914802663E-4
Population_density: 4.2426191141852057E-4
Spain: 4.161112565938298E-4
Record_label: 4.155812951033454E-4
English_language: 4.093316832476535E-4
Russia: 4.0120409579584907E-4
Race_(United_States_Census)_a07d: 3.9674976866641226E-4
Music_genre: 3.769389064131163E-4
1998: 3.7385867496300993E-4
Wiktionary: 3.648234020154463E-4
1997: 3.54699724658026E-4
Scotland: 3.5065276518587545E-4
Football_(soccer): 3.490688623067812E-4
New_York_City_1428: 3.448777362359842E-4
Wikimedia_Commons_7b57: 3.4400602687580583E-4
1996: 3.3309438577024163E-4
Sweden: 3.2934033776348527E-4
Television: 3.159235398592704E-4
1995: 3.1485577998492854E-4
California: 3.1477145413073246E-4
China: 3.112904043304386E-4
New_Zealand_2311: 3.04071166459957E-4
1994: 3.000124174542347E-4
Square_mile: 2.9850833590482203E-4
Netherlands: 2.9484295962695144E-4
Census: 2.872258622587273E-4
1993: 2.833011861119747E-4
1991: 2.8266849146720593E-4
New_York_3da4: 2.802963733542409E-4
1990: 2.787085170263239E-4
Scientific_classification: 2.73197307933567E-4
Actor: 2.7145542774407175E-4
1992: 2.708245059790496E-4
Norway: 2.6887985657244386E-4
Film: 2.6612509537974184E-4
Ireland: 2.6142804118760554E-4
Public_domain: 2.60534688257521E-4
Poland: 2.6019062192503907E-4
United_States_Census_Bureau_2c85: 2.5754277138340863E-4
Population: 2.538131376751097E-4
1989: 2.5238963001939717E-4
Marriage: 2.500861215799405E-4
Brazil: 2.489513527048767E-4
Mexico: 2.48932742474333E-4
1980: 2.462093714410407E-4

68 Album: 2.4512131323933806E-4
69 Politician: 2.450096629474081E-4
70 January_1: 2.3949795017591855E-4
71 1986: 2.3760365318440914E-4
72 Km²: 2.3633655218191835E-4
73 Record_producer: 2.3513399787970256E-4
74 1982: 2.336251342129923E-4
75 1981: 2.33022845393891E-4
76 1979: 2.3248543543860056E-4
77 Per_capita_income: 2.315706226785717 7E-4
78 1985: 2.3128722580251413E-4
79 Latin: 2.310462154828069E-4
80 South_Africa_1287: 2.304623302173043E-4
81 1983: 2.2928115186815885E-4
82 1974: 2.288211631592532E-4
83 Poverty_line: 2.2765725785827982E-4
84 1984: 2.2708544617617787E-4
85 1987: 2.2610900301470817E-4
86 French_language: 2.2380999495879702E-4
87 1970: 2.227316415327744E-4
88 1988: 2.226967546571642E-4
89 Personal_name: 2.2234401779732348E-4
90 Switzerland: 2.217096186803142E-4
91 1976: 2.2039301883184878E-4
92 1975: 2.177610528794292E-4
93 1969: 2.1399946201286832E-4
94 1972: 2.1385488716134547E-4
95 Paris: 2.1333005150452064E-4
96 1977: 2.1264624915651656E-4
97 Greece: 2.1198124228424519E-4
98 1978: 2.1166707712452158E-4
99 Animal: 2.1062988539461035E-4

They seem reasonable from intuition, for example, United_States, United_Kingdom_5ad7 and Canada are all top pages.