AYUSH SINGH

CS 6240: Assignment 5

Page Rank Matrix

Design Discussion

I followed the 5 step design strategy mentioned in the homework as follows:

Sample Graph: A -> B, C | B -> C | C -> A | D

Preprocessing: For starters I handled the case where the html file had char '&' which is invalid so I added '&' as per world wide web consortium. Apart from that, for a sample graph here's how we handle parsing from Map to Reduce:

1. **Input** Original Wikipedia data **not the Adjacency List**
2. **Map** handles two cases based on a line by line basis from sample graph:
    a. Emit all nodes with *maybeDangling* from the list of Page Names for a page
    b. Emit the page itself initializing it with *initPageRank*
3. **Reduce** handles three cases that are handled based on sample graph:
    a. A points to B which is a normal node
    b. A also points to C which is a dangling node
    c. B points to D but D is not present as a node in data i.e. dangling Node
4. **Output format** *PageName**\t**AdjacencyList*
5. **Returns** Total number of links in graph to be used as global counter in further jobs.

Allocate Unique ID: For starters I handled the case where the html file had char '&' which is invalid so I added '&' as per world wide web consortium. Apart from that, for a sample graph here's how we handle parsing from Map to Reduce:

1. **Input** Output from preprocessing step
2. **Map** handles two cases based on a line by line basis from sample graph:
    a. Emit all dangling nodes directed to one file
    b. Emit all nodes to another file which would be used as input in first iteration of PageRank computation
3. **Output format** *PageName**\t**UniqueColumnRowID*

Build Matrix: For starters I handled the case where the html file had char '&' which is invalid so I added '&' as per world wide web consortium. Apart from that, for a sample graph here's how we handle parsing from Map to Reduce:

1. **Input** Output from preprocessing step
2. **Cache** Output from Allocate Unique ID job and convert it to hashmap where key is pagename and value is unique column row ID
3. **Map** handles two cases based on whether the job is Row Major or Column Major and emits for each node in adjacency list along with their corresponding pagerank share i.e. 1/total number of adjacent nodes
4. **Output format** for each incoming record in mapper
    a. *Row Major: AdjacencyPageName***\t***PageName***\t***PageRankShare*
    b. *Column Major: PageName***\t***AdjacencyPageName***\t***PageRankShare*

ROW MAJOR VERSION

PageRank: My solution for RowMajor comprises only of Reduce intensive job.

1. **Input** Matrix built in previous step
2. **Cache** The two outputs maps generated from Allocate Unique ID job
3. **Map** takes in a records from matrix, splits by delimiter and outputs pagename row id as key with its corresponding column id and its contribution.
4. **Setup** This initializes a hash map with input from cache files and also computes dangling page rank share by iterating over dangling nodes and adding their corresponding iteration page rank or just the new one in case of first iteration.
5. **Reduce** For each value in a key, the reducer sums all pagerank in a row by fetching previous page rank from hashmap created in setup phase and multiplying that with its share computed in build matrix phase for each element in a row.
6. **Output format** *PageName***\t***PageRank*
7. ***Gist** of the job being, Matrix **M** stays the same all throughout with cell value as its corresponding page rank coefficient share (1/#adjNodes). The first job inputs pagerank matrix **R** output from unique ID job and keeps it in distributed cache to provide page rank of each column when requested by reducers for computing pageranks and saving it to a new **R'**. The other cache file is just a unique id matrix **D** of all dangling nodes values of those goes into the pagerank hashmap. Altogether achieving **R(t+1) = (M+D)R(t)***

COLUMN MAJOR VERSION

PageRank: A single iteration for Column Major solution requires two back to back jobs, one for updating pageRanks of each cell in a column and the other to reduce sums all page ranks for each cell in a row and apply page rank formula to it. The driver takes multiple inputs, one is Matrix **M** and the other being page name maps instead of cache since each column needs all its row to multiple page ranks.

1. **Matrix Map** emits nodes from matrix along with column as key and row, page rank
   Page Map emits nodes based on certain conditions
   a. If its first *iteration* means it is fresh from parser, so set it to 1/*totalLinksCount*
   b. Else emit with incoming *PageRank* from previous iteration
2. **Reduce** handles three cases that are handled as follows:
   a. Check if value is from first iteration and initialize it with new page rank.
   b. For the rest of incoming page ranks, multiply contribution with current page rank and add to page rank initialized in step a above and updated local map.
   c. In cleaup phase, I emitted the computed page ranks by iterating over hash maps.
3. **Output format** *PageName***\t***PageRank*
4. ***Phase 2 Map simply emits values it gets*** *with PageName as key and Rank as value*
5. ***Phase 2 Setup*** *is similiar to phase in row major version and is used to compute dangling node page rank share.*
6. ***Phase 2 Reduce*** *simply sums up values sent to reducer and apply page rank formula on it before emitting.*
7. **Output format** *PageName***\t***PageRank*

TopK Sort: I based off my sorting from the TopK design pattern mentioned from the book map reduce design patterns because I liked the idea of reducing the number of nodes emitted from mapper itself taking off a huge load from the reducers. On both mapper and reducer I initialized a global TreeMap which only keeps 100 values, in the cleanup phase of Mapper emit all values with a single key and repeat the same in reducer except we would not need the cleanup phase and can directly print the result. Since TreeMap is sorted we automatically get to top 100 values eventually.

Performance Comparison

Steps	6 Machines			11 Machines		
#	**List**	**Matrix Row**	**Matrix Col**	**List**	**Matrix Row**	**Matrix Col**
1 & 2	38m29s	33m19s	31m06s	22m7s	16m50s	16m17s
3	25m17s	24m11s	40m10s	13m56s	18m40s	32m03s
4	32s	1m10s	1m22s	59s	1m3s	1m5s

**Program works with the original input files.*

All matrices are generic mapreduce output files rather than sequence files.

Dangling nodes are stored in a separate file as depicted in homework solutions as matrix **D** and is fetched into distributed cache as per needs. According to my intuition, storing it separately would handle sparsity in original matrix **M**.

All nodes in matrix **M** based on whether row or column major are stored in the form of key as row/col ID and values col/row ID along with their individual pagerank contributions.

The first three phases as expected show almost double the speedup but speed-up of Top-100 for 11 machines is half that of 6 machines and after comparing stats from logs is that with 11 machines the numbers of launched map tasks were 19 compared to 9 of 6 machine which means 19*100 records would be emitted from the mapper in total compared to 9*100 of 6 machines which in turn leads to merging of 19 map outputs which eats up lot of time. The way I implemented col major partition, I had expected it to be a lot slower compared to row major which is just one job compared to multiple inputs jobs back to back with another reducer job, also looking the logs there's a lot of data shuffling.

A sample run of graph mentioned in the beginning of the document in julia.

```
In [2]: M = [0 0 1.;.5 0 0;.5 1. 0]
        D = .25
        R = .25 * [1;1;1]
        println("Dangling Share", ": ", D)
        println("PageRank Matrix", ": ", R)
        for i=1:3
            D *= .25
            R = (M*R)+D
            println(i, " ", D, " ", R)
        end

        Dangling Share: 0.25
        PageRank Matrix: [0.25,0.25,0.25]
        1 0.0625 [0.3125,0.1875,0.4375]
        2 0.015625 [0.453125,0.171875,0.359375]
        3 0.00390625 [0.363281,0.230469,0.402344]
```

Top 100 Pages

Sorted from highest ranking to lowest

The answers seems reasonable with an average difference of +-5% in page rank values compared to adjacency list implementation.

SIMPLE WIKIPEDIA DATA

United_States_09d4 0.007796874398649043

Country 0.005558113346434455

Wikimedia_Commons_7b57 0.0054579608558990925

Week 0.00386541347461503

Earth 0.003644085571355671

Water 0.0035667861352437643

Europe 0.0035412831873752717

United_Kingdom_5ad7 0.0033266266121227453

Sunday 0.0031611298617552834

Monday 0.0030995870685229087

Wednesday 0.00306217600337058

Animal 0.002980229618965195

Friday 0.0029793990118526905

Saturday 0.0029451808580120134

Thursday 0.002903567147558662

Tuesday 0.002884982785542227

France 0.0028133068497686

Asia 0.0027992783822286278

index 0.0027943812314302065

Day 0.0027885903735920645

City 0.002692644514619647

England 0.002518714749799262

Germany 0.0024335285886085047

Money 0.002425391032686868

Government 0.0023461009978964606

Number 0.0022949790624643457

Plant 0.0022443430145173402

English_language 0.0022216679227616467

India 0.0021380613902536516

Energy 0.002084367084970095

Wiktionary 0.002079341911208238

Sun 0.0020644966708373235

Italy 0.0020509895196793434

Computer 0.0020054507556035123

Wikimedia_Foundation_83d9 0.001896194686465909

People 0.0018727967885791932

Canada 0.0018335341441728648

Science 0.001781636714241836

Human 0.0017693639648917858

Spain 0.0017182446437318635

Planet 0.0017140949936970576

China 0.0016755068071997042

Japan 0.001651521459143992

State 0.0016057814803233207

Year 0.001584632810008869

Australia 0.0015797011666830096

Food 0.0015741676055625086

Mathematics 0.0015678522174881772

Russia 0.0014933989515600798

Wikipedia 0.0014904911055109427

Capital_(city) 0.0014887734251178943

Greek_language 0.0014502602719468586

Geography 0.0014069459601438843

Language 0.0013716296379888

Atom 0.0013452578510781345

Metal 0.001333759650901854

Society 0.0013235580755846146

Liquid 0.0013163233311207607

Africa 0.001309869754384879

Greece	0.0013030485012523651
Sound	0.0012962218353438448
World	0.0012677606077269611
Scotland	0.0012587036887843235
Law	0.0012379391042022928
Religion	0.0012329346421675793
Television	0.0012328190037931641
Moon	0.0012232694512845793
Light	0.0012221415254777984
Scientist	0.0012149289214856613
Culture	0.0012100856504383877
History	0.001209608743774478
2004	0.001207146384646752
Cyprus	0.0011853987242547441
Turkey	0.0011751664679987824
Plural	0.0011737999960439057
20th_century	0.0011442906590773067
Latin	0.0011308618918969013
Music	0.0011219585696889546
Poland	0.001117515512227847
19th_century	0.001093696235737468
Sweden	0.0010930490016127082
Gas	0.0010855422756429476
War	0.0010822592962129915
Information	0.0010809880779657794
Circle	0.0010800747389641916
Ocean	0.0010728378241274972
Building	0.0010632963483733382
Denmark	0.0010364514653395773
Portugal	0.0010357032537125957
Solid	0.001034148537253418
Chemical_element	0.0010226199101632377
London	0.0010186890108440083

Nation   0.0010155049444217897

Trade   0.0010038385798562814

Electricity   0.0010020645650082199

Austria   9.855972290085373E-4

Continent   9.839068201153126E-4

God   9.739018246577304E-4

Image   9.667276072907401E-4

Netherlands   9.638854026834526E-4

BIG WIKIPEDIA DATA

2006   0.0033416878297304565

United_States_09d4   0.0033149388339429172

United_Kingdom_5ad7   0.0016511016302887987

2005   0.0014820552607753377

France   0.0011268496114642155

2004   0.0010143574133664927

England   0.001009697278295116

Canada   0.001000020908016583

Germany   9.276946556061248E-4

Australia   8.294815993933588E-4

2003   8.132829627162415E-4

Japan   7.618278749689229E-4

Biography   7.577460733066671E-4

India   7.233186980919085E-4

Italy   7.070264315453115E-4

Geographic_coordinate_system   6.994392120381722E-4

2002   6.372308524887286E-4

Europe   6.350632317169174E-4

2001   6.327148060529647E-4

World_War_II_d045   6.198229692993701E-4

English_language   6.123095112761148E-4

2000   5.954897691492352E-4

London   5.782037178736105E-4

Spain 5.609408364539489E-4

Wikimedia_Commons_7b57 5.554350076838191E-4

Russia 5.50098081278838E-4

Internet_Movie_Database_7ea7 5.395446454229408E-4

Wiktionary 5.37172385926897E-4

1999 5.355594793452083E-4

Race_(United_States_Census)_a07d 5.0288257569427E-4

Population_density 4.8609842612885585E-4

1998 4.598246434150264E-4

New_York_City_1428 4.540679286161809E-4

1997 4.44534718408263E-4

Scotland 4.2884710514731826E-4

1996 4.1521375524656445E-4

Netherlands 4.0226965246671593E-4

China 4.012151332245288E-4

1995 3.943381987992303E-4

Sweden 3.93513808308884966E-4

Record_label 3.912979682776565E-4

1994 3.779903233047839E-4

January_1 3.72977028759133345E-4

1991 3.71209366254184E-4

Latin 3.6799074345089883E-4

Square_mile 3.665651103066147E-4

California 3.65172085244443327E-4

New_Zealand_2311 3.6211090957651026E-4

Television 3.620772524155732E-4

1990 3.617092121161417E-4

1993 3.5588231276986813E-4

French_language 3.506384900243668E-4

1992 3.4390268605731435E-4

New_York_3da4 3.3764268484932333E-4

Census 3.33883360204436E-4

Public_domain 3.3102740584066035E-4

Sexagenary_cycle 3.293065270170476E-4

1989 3.276029131087143E-4

Football_(soccer) 3.274634905998723E-4

1980 3.2569261056620706E-4

Ireland 3.229808931324602E-4

Music_genre 3.227514186949359E-4

Poland 3.1927920584062623E-4

Soviet_Union_ad1f 3.1833096524323465E-4

index 3.156819080258266E-4

1986 3.1475234334414897E-4

1979 3.1177288484073517E-4

1974 3.117587850662623E-4

1945 3.084165642986383E-4

1970 3.0748387621301723E-4

Norway 3.0661989541619513E-4

United_States_Census_Bureau_2c85 3.0659506811714024E-4

1981 3.06289527644558E-4

Mexico 3.061998166315452E-4

Population 3.0586792750593E-4

1982 3.03833695459014E-4

1985 3.036862386437072E-4

Switzerland 3.0217418436206196E-4

1976 3.0032277289011053E-4

Egypt 2.9882382447056325E-4

Film 2.974712191723786E-4

1969 2.9746284236335585E-4

1975 2.9725008128136645E-4

1984 2.956104440222115E-4

1983 2.9475447235354857E-4

1987 2.9288441330403237E-4

Greece 2.926436068847788E-4

Brazil 2.9258591652827763E-4

South_Africa_1287 2.920439658164599E-4

1972 2.9167618298316377E-4

Paris 2.910350266358149E-4

Gregorian_calendar 2.890538285675553E-4

Portugal 2.869973331477331E-4

Greek_language 2.8655932721647254E-4

1988 2.8616247096059964E-4

Austria 2.845295652416827E-4

1977 2.843662318480032E-4

1973 2.839223047630416E-4

1971 2.8197696067199597E-4

Denmark 2.8091020310010156E-4