COL106: Data Structures

Assignment 2

Instructor: Prof. Saroj Kaushik *Deadline: 20/02/18 11:55 pm*

1 Statement

After the first few classes of data structures, you are determined to build something big. Inspired by the story of Sergey Brin and Larry Page, founders of Google, you plan to implement a part of a Search Engine. You are not sure of what a search engine uses but have heard a bit about pagerank. You read up some articles and figure out that pagerank is a link-analysis algorithm, named after Larry Page, that assigns numerical weights to web documents. It is used in determining the relevance of a web document. Of course it is only a part of the story when it comes to Google, but nevermind. You ask some of your seniors to build the rest in the pipeline while you will handle the pagerank part. In this assignment, you are required to implement only a part (i.e. some requisites) of what may be called Vanilla Pagerank.

Pagerank maintains information about connectivity of webpages through a web matrix. Imagine there are N web pages, then you can of course maintain a **matrix M (NxN matrix)** that stores which pages have links to which other webpages. For example, M[i,j] would represent normalised number of links from the j-th webpage to the i-th webpage. Here, normalised meaning the following: Say there are n links going out of i-th webpage. Then each of the pages that have a link from i-th page get 1/n weight. For the purpose of this assignment, we do not normalise the weights, yet. We only store the count of the links.

However, N in real life can be extremely large. There were over a billion websites as of 2014. Also, commonly a website has links to only a few others website. So, most of the entries in our web matrix will be 0 and so storing all of it would be insane and a waste of space. We need a better representation for this sparsely populated matrix. In this assignment, you are required to implement a **Sparse Matrix** with **Circular Multi-Linked List** as discussed in the class (course registration example) and support a few other operations.

You need to support the following operations:

- **AddLink** <from-page-i> <to-page-j>: Increment M[j][i] by 1. If node for M[j][i] does not already exist, create a new node for M[j][i] in the multilist and initialize it with 1.
- **DeleteLink** <from-page-i> <to-page-j>: Reduce the value M[j][i] by 1. If after the operation the value reduces to 0 then remove the node corresponding to M[j][i] in the multi list. Ignore if it is M[j][i] is 0 (i.e. no node in the multi list for it).
- **RetrieveValue** <from-page-i> <to-page-j>: Return the number of links from page i to j i.e. M[j][i].
- **RetrieveRowSumUptoKthColumn** <page-i> <k>: Return sum of values first k columns in the row i in matrix M up (i.e. M[i][0...k-1]).
- **RetrieveColumnSumUptoKthRow** <page-i> <k>: Return sum of values first k rows in the column i in matrix M up (i.e. M[0...k-1][i]).
- **MultiplyVector** <n> <n-space-separated-value>: Multiple the (n x n) sub matrix of M having upper leftmost element as M[0][0] (i.e. M[0...n-1][0...n-1]) with the given vector (n x 1).

Additional operation for practice: **(Will not be evaluated)**

- **FindPageWithKthHighestOutwardLinks <k>**: Return the page number with k-th highest count of outward links. If there are multiple such pages, return the one with the smallest page number
  You must be able to support this operation in O(k) at the time of query i.e. the number of operations needed, when queried for the highest outward linking page, must be independent of the total number of webpages. (Hint: You might have another pointer in the multi-list of head pointers and keep this one sorted in descending order of outward link so as to help you traverse list of head pointers in the sorted order (decreasing order) of outward links. You might update these pointers during AddLink and DeleteLink operations)

2 Input Format

- All IO operations are to be done using standard input/output.
- The first line of the input contains a single integer representing the number of operations q
- Next q lines consist of 1 operations each in the following format:

<operation-code> <params>

Code	Operation
1	AddLink
2	DeleteLink
3	RetrieveValue
4	RetrieveRowSumUptoKthColumn
5	RetrieveColumnSumUptoKthRow
6	MultiplyVector
7	FindPageWithKthHighestOutwardLinks*

* -> This operation will not be a part of the testcases used during the evaluation

3 Output Format

- There is no output for these operator codes: 1, 2. All other operations must generate **exactly 1 line of output**.
- These operations have a single value output: 3, 4, 5, 7
- Output for MultiplyVector <n> would be n space separated integers representing the result of the matrix vector multiplication. (No space after the last entry from result)

4 Limits

- $N = 1 \times 10^9$
- 64 bit integer datatype would be sufficient for all calculations
- For MultiplyVector operation, $n \leq 10^6$
- For other operations, $0 \leq i, j < N$, $1 \leq k \leq N$

5 Sample IO

5.1 Input

```
10
1 0 1
1 0 2
1 3 4
1 0 1
3 0 4
4 1 100
5 0 100
2 0 2
5 0 100
6 5 1 2 3 4 5
```

5.2 Output

```
0
2
3
2
0 2 0 0 4
```

6 Submission Instructions

- The program file should be named with your Kerberos ID. For example, if your kerberos id is cs1150999 then the file name should be cs1150999.cpp or cs1150999.c.

7 Other Instructions

- The assignment is to be done individually.
- All submissions must use C/C++ for the programming assignment.
- You are **not** allowed to used any standard implementations/libraries (except for IO) in any part of the program.
- This assignment has been designed to enforce use of linked lists over arrays so the limits are very high. Remember that you would not be able to allocate memory of sizes $O(10^9)$. So you will have to be smart with your memory allocations and avoid any unnecessary memory usages.
- You are not allowed to discuss or take help from anybody outside the class. You may only discuss but are not allowed to share code with anyone inside the class.
- We will run plagiarism detection on your submissions. People found guilty will be penalised as per the instructions mentioned in the beginning of the course