Alexandria University
Faculty of Engineering
Computer and Systems Engineering Dept.
Fourth Year
Spring 2016

CS432: Distributed Systems and Net-Centric Computing
Assigned: Monday 7th Mar, 2016
Due: Sunday 20th Mar, 2016

Incremental calculation of shortest path in dynamic graphs.

Objective

This assignment aims to make you think more about distributing work load and utilize the available machines in the best way possible.

Description

In graph theory, the shortest path problem is the problem of finding a path between two vertices (or nodes) in a graph such that the sum of the weights of its constituent edges is minimized. This is a fundamental and well-studied combinatorial optimization problem with many practical uses: from GPS navigation to routing schemes in computer networks; search engines apply solutions to this problem on website interconnectivity graphs and social networks apply them on graphs of peoples' relationships.

In this project, the task is to answer shortest path queries on a changing graph, as quickly as possible. We will provide an initial graph which you may process and index in any way you find necessary. Once this is done, we will begin issuing a workload consisting of a series of sequential operation batches. Each operation is either a graph modification (insertion or removal) or a query about the shortest path between two nodes in the graph. Your program is expected to correctly answer all queries as if all operations had been executed in the order they were given.

The graphs are directed and unweighted. Input to your program will be provided via standard input, and the output must appear on the standard output.

Specifications

There are three operations that are applied to the original graph. The three operation types are as follows:

- **'Q'/query:** this operation needs to be answered with the distance of the shortest (directed) path from the first node to the second node in the current graph. The answer

should appear as output in the form of a single line containing the decimal ASCII representation of the integer distance between the two nodes, i.e., the number of edges on a shortest directed path between them. If there is no path between the nodes or if either of the nodes does not exist in the graph, the answer should be -1. The distance between any node and itself is always 0.

- **'A'/add:** This operation requires you to modify your current graph by adding another edge from the first node in the operation to the second. As was the case during the input of the original graph input, if the edge already exists, the graph remains unchanged. If one (or both) of the specified endpoints of the new edge does not exist in the graph, it should be added. This operation should not produce any output.
- **'D'/delete:** This operation requires you to modify your current graph by removing the edge from the first node in the operation to the second. If the specified edge does not exist in the graph, the graph should remain unchanged. This operation should not produce any output.

Each operation is represented by one character ('Q', 'A' or 'D') that defines the operation type, followed by a space and two positive integer numbers in decimal ASCII representation, also separated by a space. The two integer numbers represent node IDs.

Sequence

- First the initial graph is entered to your program's standard input. The graph is represented as a list of edges, each of which consists of a pair of node ids (the edge starting node, followed by the edge destination node) represented as non-negative integer numbers. Your program will receive multiple lines (each representing one edge) containing exactly 2 integer numbers in decimal ASCII representation separated by a single space. The initial graph ends with a line containing the character 'S'.

1. Your program's standard output should type a line containing the character 'R' (case insensitive, followed by the new line character '\n'). Your program uses this line to signal that it is done ingesting the original graph, has performed any processing and/or indexing on it and is now ready to receive the workload.

2. The workload is delivered in batches. Each batch consists of a sequence of operations provided one per line followed by a line containing the single character 'F' that signals

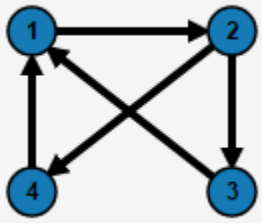
the end of the batch.

3. After the end of every batch, we will wait for output from your program before providing the next batch. You need to provide as many lines of output as there are query ('Q') operations in the batch - each line containing the distance of the shortest path as described above. Your program is free to process operations in a batch concurrently. However, the query results in the output must reflect the order of the queries within the batch.

Examples

The initial graph is entered as follows

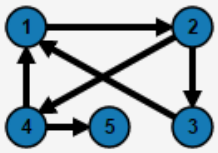
```
1 2
2 3
3 1
4 1
2 4
S
```



The following batches are then added to the graph

Batch 1:

```
Q 1 3
A 4 5
Q 1 5
Q 5 1
F
```

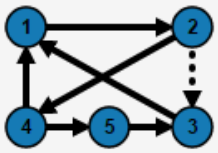


Output:

```
2
3
-1
```

Batch 2:

```
A 5 3
Q 1 3
D 2 3
Q 1 3
F
```



Output:

```
2
4
```

Notes

- This assignment is from the ACM Sigmod Programming contest 2016. http://dsg.uwaterloo.ca/sigmod16contest/task.shtml

- In order to make your program efficient it is advisable to use C programming language, However, you can use the language you choose and try to use all the optimization techniques to increase performance.
- It is highly recommended that you follow the submitting rules in the contest page.

However, for this project the required is that your program correctly follows the steps above.

- It is also highly recommended that you consider participating in the contest, be sure that you will be provided by advice, guidance and support needed to participate.
- Your solution will be evaluated for correctness and execution time. Execution time measurement **does not start until your program signals (with 'R') that it is finished ingesting the intial graph.** Thus, you are free to pre-process or index the graph as you see fit without penalty, as long as your program runs within the overall testing time limit. Concurrent request execution within each batch is allowed and encouraged, as long as the results mimic a sequential execution of the operations within the batch. In particular, the result for each query must reflect all additions and deletions that precede it in the workload sequence, and must not reflect any additions and deletions that follow it.
- You should deliver a report, please follow the template.
- Please reuse your code submitted for the first assignment whenever possible.

Grading Policies

- No Late submission is allowed.
- You should work in groups of 3 students.
- Plagiarizing is not acceptable. Sharing code fragments between groups is prohibited and all the groups that are engaged in this action will be severely penalized. Not delivering the assignment will be much better than committing this offence.