Programming Design, Spring 2014
Homework 8

Instructor: Ling-Chieh Kung
Department of Information Management
National Taiwan University

Submission. To submit your work, please upload the following file to the online grading system at http://lckung.im.ntu.edu.tw/PD/.

1. Your .cpp file for Problem 1.

Each student must submit her/his individual work. No hard copy. No late submission. The due time of this homework is ***8:00am, April 28, 2014***. Please answer in either English or Chinese.

Problem 0

(0 point) Please read Chapter 11 of the textbook.[1] In any case, I strongly suggest you to read the textbook thoroughly before you start to do this homework.

Problem 1

(100 points) Your apple delivery business is expanding. How to efficiently deliver apples to customers depends on operations on the network you built last time. Therefore, in this homework we will try to make your `Network` class more powerful by adding several overloaded operators into it.

One task that sometimes bothers you is to compare different networks. Recall that each network represents a collection of arcs (roads) and nodes (intersection of roads). In some cases, you are given two networks and you would like to see whether one includes the other. More precisely, network $G_1 = (V_1, E_1)$ includes $G_2 = (V_2, E_2)$ if $V_2 \subseteq V_1$ and $E_2 \subseteq E_1$. As an example, the network in Figure 1 includes that in Figure 2. When we compare two networks, arcs are treated as the same as long as their sources and destinations are the same, even if their weights are different.

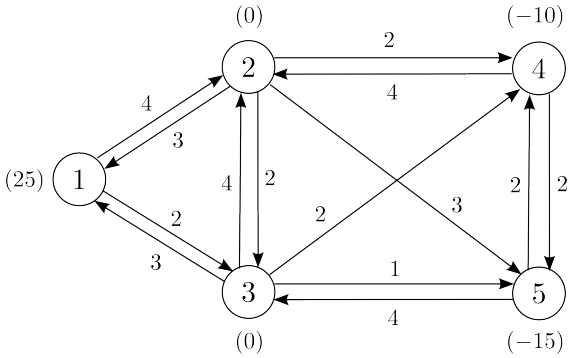

Figure 1: Network G_1

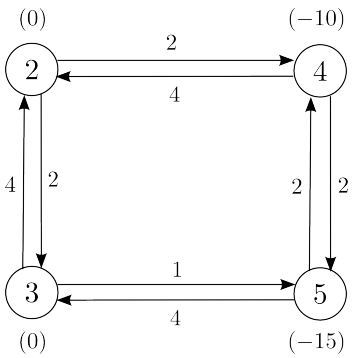

Figure 2: Network G_2

Other tasks that you may want to perform are addition and subtraction between networks. Suppose we are given two networks $G_3 = (V, E_3)$ and $G_4 = (V, E_4)$ with the same set of nodes V, $G_3 + G_4 = (V, E_3 \cup E_4)$ still has the same set of nodes. Moreover, the set of arcs $E_3 \cup E_4$ is the union of the two original arc sets. What if E_3 and E_4 both contain an arc (u, v) with different weights w_{uv}? In this case, the weight of arc $(u, v) \in E_3 \cup E_4$ is the *smaller* one. Let's take the two networks in Figures 3 and 4 as an example. For these two networks, their sum is the network in Figure 5. Please note that while arc $(3, 2)$ exists in both the two networks in Figures 3 and 4, in Figure 5 its weight is 4 because $4 < 8$.

[1] The textbook is *C++ How to Program: Late Objects Version* by Deitel and Deitel, seventh edition.

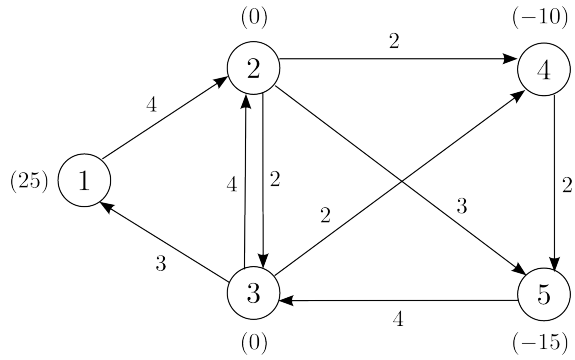

Figure 3: Network G_3

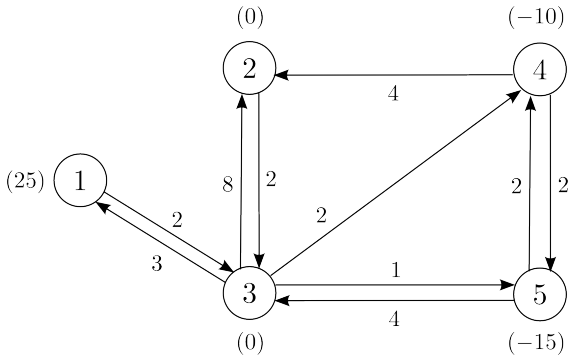

Figure 4: Network G_4

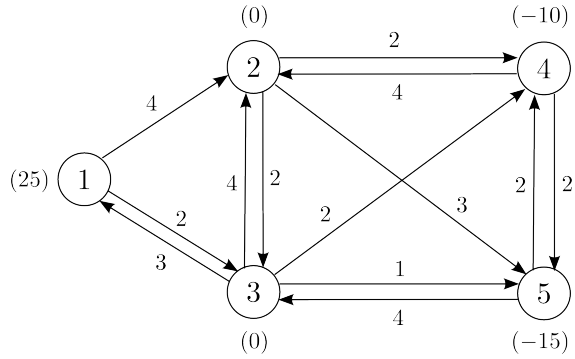

Figure 5: Network $G_3 + G_4$

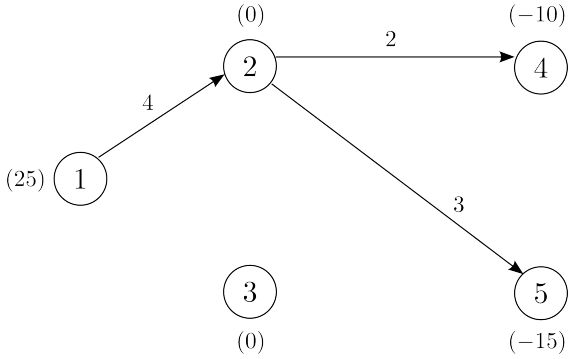

Figure 6: Network $G_3 - G_4$

How about subtraction? Still given $G_3 = (V, E_3)$ and $G_4 = (V, E_4)$ with the same set of nodes V, the network $G_6 = G_3 - G_4 = (V, E_3 \setminus E_4)$. More precisely, we have $(u, v) \in G_3 - G_4$ if and only if $(u, v) \in G_3$ and $(u, v) \notin G_4$, even if the two arcs have different weights. For example, the outcome of subtracting the network in Figure 3 by that in Figure 4 is that in Figure 6.

Please note that we do not define addition and subtraction between networks with different sets of nodes. In this case, the operation simply fails.

The class

Last time you have built a class `Network`. For this homework, please overload four operators: `<=`, `>=`, `+`, and `-`. For the first two operators, the parameter is another `Network` and the returned type should be `bool`; for the last two, the parameter is another `Network` and the returned type should be `Network` which may be put at the left-hand side of an assignment operator. In any case, the argument should not be modified.

Besides operator overloading, you also need to implement a copy constructor for `Network`. Of course, deep copy is required.

Input/output formats

The input will consist of several lines of characters and integers. In each line, first there is one or two characters indicating the task to be done for this line. Following integers then provide information regarding that task. Two consecutive items are always separated by a white space. There are ten tasks in total, where the first five are almost identical to the five in the previous homework. The only difference is that one addition integer is used to indicate the network that the task should apply to. The ten tasks are:

- (Adding a node) When a line starts with `AN` followed by three integers i, v, and w, one should try to add a node whose id is v and weight is w into network G_i. The node should be added if and only if no existing node uses v as its id in G_i.

- (Adding an arc) When a line starts with `AA` followed by four integers i, u, v, and w, one should try to add an arc from node u to node v with weight w into network G_i. The arc should be added if and only if no existing arc goes from u to v in G_i.

- (Removing a node) When a line starts with `RN` followed by two integers i and v, one should try to remove the node whose id is v from network G_i. Once the node is removed, all arcs having it as an endpoint should be removed from G_i. Nothing should happen to G_i if and only if no node uses v as its id in G_i.

- (Removing an arc) When a line starts with `RA` followed by three integers i, u, and v, one should try to remove the arc going from u to v from network G_i. Nothing should happen to G_i if and only if no such arc exists in G_i.

- (Is there a path?) When a line starts with `P` followed by a sequence of integers G_i, v_1, v_2, ..., and v_n, one should check whether a path $(v_1, v_2, ..., v_n)$ exists in G_i. If yes, one should output two integers, the total length of this path followed by the sum of supply quantities of all nodes on this path. Of course, a white space is used to separate the two output numbers. If no such path exists in G_i, simply output one integer 0. In the given path, a node or an arc may be passed by more than once. In this case, a node's supply quantity should be counted only once but an arc's distance should be counted as many times as it is passed by.

  Note. If G_i does not exist, output 0.

- (Including another network?) When a line starts with `<=` followed by two integers i and j, one should output 1 if G_i is included in G_j or 0 otherwise. If G_i or G_j does not exist or if their set of nodes are different, output 0.

- (Included in another network?) When a line starts with `>=` followed by two integers i and j, one should output 1 if G_i includes G_j or 0 otherwise. if G_i is included in G_j or 0 otherwise. If G_i or G_j does not exist or if their set of nodes are different, output 0.

- (Adding two networks) When a line starts with `+` followed by two integers i and j, one should modify G_i to $G_i + G_j$ and leave G_j unchanged.

- (Subtracting one network from another) When a line starts with `-` followed by two integers i and j, one should modify G_i to $G_i - G_j$ and leave G_j unchanged.

- (Copying a network) When a line starts with `C` followed by two integers i and j where G_i exists but G_j does not, a new network G_j should be created to be exactly the same as G_i. If G_i does not exist or G_j already exists, do nothing.

Node weights may be nonpositive while all other integers are all positive. For tasks `P`, `<=`, and `>=`, some things should be output.

As an example, if "PDSp14_hw08_input.txt" is the input file, "PDSp14_hw08_output.txt" contains what should be output by your program. From `AN 1 1 25` to `AA 1 5 4 2` we build network G_1 in Figure 1. We then copy G_1 to create G_3. After removing arcs by doing tasks from `RA 3 1 3` to `RA 3 5 4`, we get G_3 in Figure 3. We then copy G_1 to create G_4, modifying G_4 by doing tasks from `RA 4 1 2` to `AA 4 3 2 8`, and getting G_4 in Figure 4. As G_1 includes G_3 and G_1 includes G_4 (even if for arc $(3, 2)$ the weights are different), we output two 1s; then as neither G_3 nor G_4 include the other, we output two 0s. The addition and subtraction tasks then results in networks in Figures 5 and 6. For G_5, the one depicted in Figure 6, no arc goes from node 1 to node 3. Therefore, we output 0 for `P 5 1 3`.

Note. To make the homework easier, let's restrict the maximum number of networks to be 10. Moreover, they will be labeled as G_1, G_2, ..., and G_{10}. If you think that helps, you may add a private member variable into `Network` to record the network ID.

Grading criteria

- 70 points are given based on the correctness of your output. Each correct line of output gives you two points. The input will be organized in the following way:
  - In the first part, there are only `AN`, `AA`, `RN`, and `RA` tasks. Then ten `P` tasks will be assigned to test your functions for adding nodes and arcs into various networks. Please note that conflicts are possible.
  - In the second part, there will be five `<=` or `>=` tasks for networks constructed in the first part.
  - In the third part, there will also be `+` or `-` tasks for networks constructed in the first part. Then ten `P`, `<=`, or `>=` tasks will be assigned.
  - In the last part, there will be some `C` tasks. Then ten `P`, `<=`, or `>=` tasks will be assigned.
- 30 points will be based on how you write your program, including the logic and format. Please try to write a robust, efficient, and easy-to-read program.

  Note. If you do not implement operator overloading and a copy constructor, you will get very low grades for this part.