Homework #4

Due Time: 2015/12/21 (Mon.) 12:00
Contact TAs: `ada@csie.ntu.edu.tw`

Instructions and Announcements

- For the first time submitting your ADA 2015 HW, please create a repository for ADA on bitbucket (`https://bitbucket.org`) and share the repository with user ada2015 with read permission.
- You have to login to the judge system (`http://ada2015.csie.org`) to bind the bitbucket repository to your account. For programming problems (those containing "**Programming**" in the problem topic), push your source code to your bitbucket repository. After you push the source code to repository, you can run the judge and get the score. Please refer to the guide of the judge system on its index page for further details.
- For other problems (also known as "hand-written problems"), submit the answers in an electronic copy via the git repository before the due time. Please combine the solutions of all these problems into **only ONE file** in the PDF format, with the file name in the format of "`hw[# of HW].pdf`" (e.g. "`hw1.pdf`"), all in **lowercase**; otherwise, you might only get the score of one of the files (the one that the grading TA chooses) or receive penalty because of incorrect filename.
- Discussions with others are strongly encouraged. However, you should write down your solutions `in your own words`. In addition, for **each and every** problem you have to specify the references (the Internet URL you consulted with or the people you discussed with) on the first page of your solution to that problem. You may get zero point for problems with no specified references.
- **NO LATE SUBMISSION ONE DAY AFTER THE DUE TIME IS ALLOWED.** For all submissions, up to one day of delay is allowed; however, penalty will be applied to the score according to the following rule (the time will be in seconds):

$$\text{LATE\_SCORE} = \text{ORIGINAL\_SCORE} \times (1 - \text{DELAY\_TIME}/86400)$$

  Note that late submission of partial homework is NOT allowed. The penalty will be applied to the entire homework in the case of late submission.

Problem 1 (18%)

We have learned several graph traversal algorithms, including BFS and DFS, in class. Now let's turn our focus on three tree traversal algorithms: pre-order traversal, in-order traversal, and post-order traversal. Given a tree and a tree traversal algorithm, it is easy to create the sequence of traversed nodes. But how about the other way around? Can you reconstruct a binary tree that satisfies given traversal sequences?

Please design an algorithm to construct a legal binary tree with the giver traversal lists.

Hint:

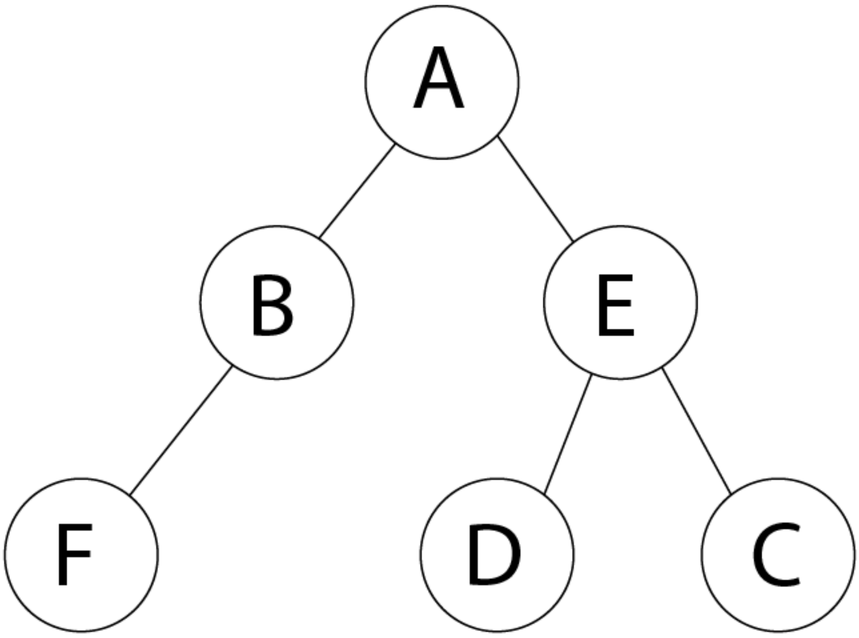

Figure 1: Example

The pre-order traversal list of Figure 1 is {A, B, F, E, D, C}.

The in-order traversal list of Figure 1 is {F, B, A, D, E, C}.

The post-order traversal list of Figure 1 is {F, B, D, C, E, A}.

(1) (6%) Given a pre-order traversal list $\{pre_1, pre_2, pre_3, ..., pre_n\}$ and a post-order traversal list $\{post_1, post_2, post_3, ..., post_n\}$. Please design an algorithm to reconstruct the binary tree that satisfies the two traversal lists. The time complexity of your algorithm should be under O(N^3).

(2) (6%) Given a pre-order traversal list $\{pre_1, pre_2, pre_3, ..., pre_n\}$ and an in-order traversal list $\{in_1, in_2, in_3, ..., in_n\}$. Please design an algorithm to reconstruct the binary tree that satisfies the two traversal lists. The time complexity of your algorithm should be under O(N^3).

(3) (6%) Given a in-order traversal list $\{in_1, in_2, in_3, ..., in_n\}$ and a post-order traversal list $\{post_1, post_2, post_3, ..., post_n\}$. Please design an algorithm to reconstruct the binary tree that satisfies the two traversal lists. The time complexity of your algorithm should be under O(N^3).

Problem 2 (22%)

Alice, who lives in the wonder-bitland, has the magic power to eliminate two numbers at once if there is exactly one different bit between the two numbers in binary representation. In addition, she can also use the magic power to eliminate any arbitrary number too.

Given N different integers, please tell Alice the minimum number of magic power used to eliminate all the N integers.

Hint: Alice only needs to use magic power once to eliminate $\{2, 6\}$ because $(10)_2$ and $(110)_2$ differ by exactly one bit, but will have to use magic power twice to eliminate $\{1, 6\}$ since there are three different bits between $(1)_2$ and $(110)_2$.

(1) (6%) Please design an algorithm to separate the numbers into two groups such that Alice can't eliminate two numbers at once in either group.

(2) (8%) Please describe what Edmonds-Karp algorithm is and briefly justify its time complexity. *Hint: Read Chapter 26 of Introduction to Algorithm, 3ed for reference.*

(3) (8%) Please design and describe an algorithm using Edmonds-Karp algorithm and the two groups you previously defined to calculate the minimum number of magic power used to eliminate all the N integers.

Problem 3 (30%)

In this problem, we will find the ***minimum mean cycle*** of a weighted directed graph $G = (V, E)$.

For a cycle $c = e_1, e_2, \cdots, e_k$, its *mean weight* is the average of weights of its edges:

$$\mu(c) = \frac{1}{k} \sum_{i=1}^{k} w(e_i)$$

And the *minimum mean cycle* is the cycle with minimum *mean weight*:

$$\mu^*(G) = \min_c \mu(c)$$

If G doesn't contain any cycle, define $\mu^*(G) = \infty$.

For simplicity, we add a source vertex s into G, and connect s to all other vertices with 0-weight edges, then all $v \in V$ can be reached from s. This won't create or affect any cycle.

If $\mu^*(G) < 0$, then it contains negative cycles, which is miserable because shortest path can't be defined. We call those G *miserable-word graph*s. On the other hand, graphs with $0 < \mu^*(G) < \infty$ is better, we call them *good-word graph*s.

But both of them are not good enough. Consider graphs with $\mu^*(G) = 0$, which is called *con-word graph*s (*con* means "zero"). In this case, *con-weight cycle*s (cycle with total weight zero) are just *minimum mean* cycles.

Let $G = (V, E)$ be a *con-word* graph, $|V| = N, |E| = M$. Let $d(v)$ be the length of shortest path from s to v; $d_k(v)$ be the shortest length among all paths from s to v passing through **exactly** k edges. If such a path doesn't exist, define the length to be ∞.

(1) (1%) Prove that G has no negative cycles.

(2) (4%)

(a) (2%) Prove that for all $v \in V$,

$$d(v) = \min_{0 \le k \le N-1} d_k(v)$$

that is, there exists a shortest path from s to v with at most $N - 1$ edges.

(b) (2%) Prove that for all $v \in V$,

$$\max_{0 \le k \le N-1} \frac{d_N(v) - d_k(v)}{N - k} \ge 0$$

If $d_N(v) = d_k(v) = \infty$ for some k, define there difference as $\infty - \infty = 0$.

(3) (8%) Let c be a *con-weight* cycle.

(a) (2%) For every edge $e \in c$, if e starts from u and end at v, prove that

$$d(v) = d(u) + w(e)$$

Hint: Consider the path from v back to u using edges of c.

(b) (5%) Prove that there exist vertex v on c such that

$$d_N(v) = d(v)$$

and hence

$$\max_{0 \leq k \leq N-1} \frac{d_N(v) - d_k(v)}{N-k} = 0$$

Hint: First, find a vertex v on c with $d_k(v) = d(v), 0 \leq k \leq N-1$. Can you find another vertex u on c with $d_{k+1}(u) = d(u)$?

(c) (1%) Prove that

$$\min_{v \in V} \max_{0 \leq k \leq N-1} \frac{d_N(v) - d_k(v)}{N-k} = 0$$

(4) (7%) Let p be a shortest path[1] from s to v with exactly N edges.

(a) (3%) Prove that p contains at least one cycle. If one such cycle has length l and total weight W, prove that

$$d_N(v) - d_{N-l}(v) \geq W$$

(b) (4%)For all $v \in V$ with

$$\max_{0 \leq k \leq N-1} \frac{d_N(v) - d_k(v)}{N-k} = 0$$

, prove that **every** shortest path[2] from s to v contains at least one *con-weight* cycle, but doesn't contain any *positive-weight* cycle.

(5) (10%) Now come back to the original problem. Consider a general graph G. G can be either *Miserable-word*, *good-word*, or *con-word*, or doesn't even contain cycles.

(a) (3%) Prove that

$$\mu^*(G) = \min_{v \in V} \max_{0 \leq k \leq N-1} \frac{d_N(v) - d_k(v)}{N-k}$$

Hint: If $\mu^*(G)$ is finite, subtract the weight of every edge by $\mu^*(G)$. How much will $d_k(v)$ change?

(b) (7%) Design an algorithm to determine $\mu^*(G)$, and if $\mu^*(G) < \infty$, **find a minimum mean cycle**. Simply explain the correctness of your algorithm and analyze its time complexity. It should run in $O(N^2 + NM)$ time.

[1]Here *shortest path* means shortest among all paths from s to v with exactly N edges.

[2]Again, among all paths with exactly N edges.

Problem 4 - MUST (Programming, 30+6%)

Description

HH just learned MST (Minimum Spanning Tree) in the ADA course. To practice, he wants to find a MST of an undirected graph G. But he notices that there may be more than one possible MST of G. To make his life easier, he tries to choose the edges exist in every MST first. We call such edges *MUST edges* since you must choose them in MST. However, finding MUST edges is a hard task, can you help him?

Input Format

The first line contains an integer T indicating the total number of test cases. Each test case starts with a line contains two integers n, m, denoting the number of nodes and edges in the undirected graph G. Each of the following m lines contains three integers a_i, b_i, c_i, which means there is an edge (a_i, b_i) with weight c_i in G.

- $1 \leq T \leq 10$
- $3 \leq n \leq 50000$
- $n - 1 \leq m \leq 100000$
- $1 \leq a_i, b_i \leq n$ and $a_i \neq b_i$
- $1 \leq c_i \leq 10^7$
- G is connected

Output Format

For each test case, please output the number of MUST edges and the sum of weight of MUST edges in one line.

Sample Input

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

Sample Output

```
1 2
0 0
2 6
```

Hint

- There are two bonus tests (3 points each) in this problem. The condition $n \leq 500$ and $m \leq 50000$ holds for the first 10 tests.