Homework #3

Due Time: 2019/12/12 (Thur.) 14:20
Contact TAs: `ada-ta@csie.ntu.edu.tw`

Instructions and Announcements

- There are **three programming problems** and **three hand-written problems**.
- **Programming.** The judge system is located at `https://ada-judge.csie.ntu.edu.tw`. Please login and submit your code for the programming problems (i.e., those containing "Programming" in the problem title) by the deadline. NO LATE SUBMISSION IS ALLOWED.
- **Hand-written.** For other problems (also known as the "hand-written problems"), you **MUST** turn in **a printed/written version** of your answers to **the submission box at your classroom**. You can also upload your homework to the NTU COOL system as a backup; however, it will be marked **only when** you have turned in the printed/written answer but it is lost during the grading process.
  NO LATE SUBMISSION IS ALLOWED.
- **Collaboration policy.** 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 (e.g., the Internet URL you consulted with or the people you discussed with) on the first page or comment in code of your solution to that problem. You may get zero point due to the lack of references.

Problem A - Idol Master! (Programming) (15 points)

Problem Description

The story. As we all know, students in the NIIU CSIE department often idolize peers whom they look up to. Due to this phenomenon, WillyPillow decides to design a game called *Idol Master!*. In the game, each character either has the character it admires the most as its idol or has no idols. In the source code, each character is represented by the following structure:

```
struct Character {
    // The character's idol, null if empty.
    std::shared_ptr<Character> idol;
    // Other data related to the character.
    CharacterData data;
};
```

Introduced in C++11, `std::shared_ptr` is a smart pointer supporting reference counting. Specifically, an instance of the pointer tracks the number of its existing copies, and destructs the referenced object if the number falls to zero.

WillyPillow has a list of characters and their corresponding idols together with a list of characters that appears in a certain scene. (Due to the nature of the scene, it is guaranteed that no character in it is the idol of another character.) However, the memory usage of his game is too high, so he needs to remove some characters from the scene. As he wants to compare how much memory he can save by removing different sets of characters, he needs to know the number of character objects that can be destructed, i.e., its corresponding character is destroyed or all character objects that link to it are destructed, if we remove a given set of characters.

Problem formulation. Formally speaking, we say a vertex with a zero in-degree is *not referenced* and shall be removed. A vertex is *destructed* if it is removed either manually or due to being *non-referenced*.

You are given a directed graph G in which the out-degree of each vertex is at most 1 [1], and multiple queries on the same graph G. Each query results in a new graph G' that is constructed by removing a set of vertices X and their outgoing edges on G. You can assume that all vertices in X have no incoming edges. Please implement a program to compute the number of vertices that are *destructed* on G' for each query.

Note that *destructing* one vertex may cause another vertex to become *non-referenced* and be *destructed* as well. In addition, if a vertex not in X already has a zero in-degree in G, then it is *not destructed* (one can imagine that they are connected to by a special hidden vertex).

Input Format

The first line contains the number of vertices, N.

The second line consists of N integers. The i-th integer, p_i ($0 \le p_i \le N$) indicates that there is an edge $i \to p_i$. Notably, $p_i = 0$ if the out-degree of i is zero.

The third line consists of the number of queries Q.

Q lines follow, the i-th of which starts with an integer x_i ($x_i \ge 1$), denoting the number of vertices to be removed, and is followed by x_i integers, denoting the set of vertices to be removed.

[1] This is known as a directed pseudoforest.

Test Group 0 (0%):

- Sample Input

Test Group 1 (15%):

- $N \leq 2^{13}$
- $\sum x_i \leq 2N$

Test Group 2 (85%):

- $N \leq 2^{18}$
- $\sum x_i \leq 2N$

Output Format

For each query, output a line indicating the number of vertices that can be deleted according to the rules above.

Sample Input 1

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

Sample Output 1

```
1
```

Sample Input 2

```
8
1 6 6 6 1 2 3 4
12
2 5 7
2 7 8
1 5
1 8
1 8
2 5 7
1 5
2 7 8
1 7
1 5
1 8
1 5
```

Sample Output 2

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

Problem B - Traveling Pony Problem (Programming) (15 points)

Problem Description

A little pony is traveling in a city. He starts from his home s and goes to the destination t. However, each road in the city has two values d_i and l_i, the length of the road and the level of danger. To ensure his safety, the pony needs to be well-prepared before the trip. He has a non-negative value L indicating the amount of protection. To pass a road safely, he needs to ensure that $L \geq l_i$. Since the protection is expensive, he wants to minimize the amount of protection. Your goal is to calculate the shortest path under the condition that a minimum amount of protection is needed.

Input

The map can be considered as an undirected graph. Each road is an edge on the graph. And the pony travels from a vertex s, which is his home, to a vertex t, which is the destination.

The first line of the input consists of 2 integers n, m, which indicate the number of vertices and the number of edges. The second line consists of 2 integers s and t, indicating pony's home and the destination. On the follow m lines, there are 4 integers x_i, y_i, d_i, and l_i, which means an edge connecting vertices x_i and y_i; the distance between x_i and y_i is d_i, and the level of danger is l_i.

- $2 \leq n \leq 200000$
- $1 \leq m \leq 500000$
- $0 \leq s, t < n$
- $0 \leq x_i, y_i < n$, $x_i \neq y_i$ for $0 \leq i < m$
- There is no two edges connecting a same pair of vertices.
- $0 \leq d_i, l_i \leq 10^9$
- The graph is connected.

Output

You need to output two non-negative integers D and L. D is the length of shortest path if the pony only goes through roads whose danger level $\leq L$; L is the minimum amount of protection needed that the pony can arrive the destination safely.

Subtask 1 (30 %)

- $l_0 = l_1 = ... = l_{m-1}$

Subtask 2 (30 %)

- $l_i \neq l_j$ for $0 \leq i, j < m$ and $i \neq j$

Subtask 3 (40 %)

- No other constraints.

Sample Input 1

```
5 5
0 4
0 1 1 3
1 2 2 1
2 3 1 2
3 4 1 3
0 4 1 4
```

Sample Output 1

```
5 3
```

Sample Input 2

```
5 5
0 4
0 1 1 3
0 2 2 2
1 3 1 4
2 3 2 2
3 4 2 5
```

Sample Output 2

```
4 5
```

Problem C - Magic Song (Programming) (20 points)

Problem Description

You have a collection of songs. Each song contains some "jump points", and when a song plays to a jump point, you can optionally jump to another song. Your goal is to determine how to jump between songs so as to maximize the length of played music.

Input

The first line of the input file contains an integer indicating T, the number test cases.

For each test case, the first line contains two integers, N and M, indicating the number of songs and the number of total "jump points", respectively. The second line contains N integers separated by spaces, $a_1, a_2..., a_i$, indicating the length of i-th song (in terms of the number of notes). The following M lines, each of which contains four integers separated by spaces, s_1, t_1, s_2, t_2, indicating that the ending of the t_1-th note of the s_1-th song could jump to the beginning of t_2-th note of the s_2-th song.

- $1 \leq T \leq 10$
- $1 \leq N \leq 10^5$
- $0 \leq M \leq 5 \times 10^5$
- $1 \leq a_i \leq 10^8$
- $1 \leq s_1, s_2 \leq N$
- $1 \leq t_1 \leq a_{s_1}$
- $1 \leq t_2 \leq a_{s_2}$

Test Group 1 (30 %)

- $1 \leq N, M \leq 10^3$
- $1 \leq a_i \leq 10^3$

Test Group 2 (30 %)

- $1 \leq N, M \leq 10^4$

Test Group 3 (40 %)

- No other constraints.

Output

Please output an integer indicating the maximum number of notes that you can play. If the music can be played forever, output "LoveLive!".

Sample Input 1

```
1
3 3
10 10 10
1 5 2 3
```

```
2 7 1 6
2 3 3 5
```

Sample Output 1

```
15
```

Sample Input 2

```
1
1 1
10
1 6 1 3
```

Sample Output 2

```
LoveLive!
```

Hint

Some example of suite:

https://www.youtube.com/watch?v=95tLMjeK0kw

https://www.bilibili.com/video/av5986635

Problem D - Basic Problems in Graphs (Hand-Written) (20 points)

1. Give one example of a directed graph with four vertices, A, B, C, and D, such that both depth-first search and breadth-first search discover the vertices in the **same order** when starting at A. Give one example of a directed graph where DFS and BFS discover the vertices in **different orders** when starting at A. ("Discover" means the time that the algorithm first reaches the vertex, when the vertex color becomes gray in CLRS textbook.) You can assume that both DFS and BFS iterate over outgoing neighbors in an alphabetical order. (4 points, 2 points for each example)

2. Prove or disprove that the minimum spanning tree is the same as the shortest path tree. Please give one example to explain your proof. (You need to show 3 graphs, 1 for the original graph, 1 for the shortest path tree on that graph, and 1 for the minimum spanning tree on that graph. Note that the number of vertices in your graph should be no greater than 5.) (4 points)

3. We are given a network of V cities and E roads. All roads are one-way, i.e., if a road goes from the city u to the city v, then there will be no roads from v to u. In addition, the roads would **not form any cycle**. That is, you cannot start from a city v, go through several cities, and go back to v. Finally, we assume that there is a capital that can go to all other cities in the network.

   We want to compute the minimum number of days for a tourist to go from the capital to each non-capital city. When a tourist encounters a city v, he will spend $c(v) > 0$ days to visit all attractions in v. When a tourist goes from the city u to the city v, he will spend $r(u, v) > 0$ days on the road. Now given the function c for all cities, and the function r for all roads, please compute the minimum number of days for a tourist to go from the capital to each non-capital city. We use $M(v)$ to denote this minimum number of days from the capital to a city v. We illustrate the problem with an example of four cities A, B, C, D and four roads. Let A be the capital and $c(A) = c(B) = c(C) = c(D) = 1$, and $r(A, B) = r(A, C) = r(B, D) = 2$, and $r(C, D) = 10$. Then $M(D) = c(A) + r(A, B) + c(B) + r(B, D) + c(D) = 7$. Similarly $M(B) = M(C) = 4$.

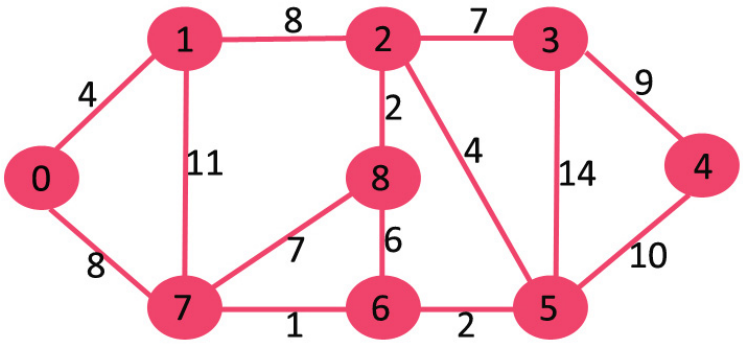
   (a) Please derive an *optimal* algorithm to compute the function M for all cities other than the capital. (6 points)

   (b) Please analyze the time complexity of your algorithm and prove that it is correct. (You need to show why your algorithm gives the minimum number of days from the capital to a city v.) (6 points)

Problem E - Mailing Fees (Hand-Written) (22 points)

Your computer bought on the Double Eleventh Day has broken, so you have to send it to a factory to repair. There are N factories and E roads, and you need to send your device to one of them. When you send it to any factory, you have to pay the mailing fees. As shown in the following graph, the starting position is the node 0; each vertex represents one factory, and an edge between two factories shows the mailing fee. You can ask a factory to transfer your computer to another one, and you still have to pay the mailing fees. Note that you need to pay the mailing fees for the factory sending back your device too. Your goal is to find out the minimum amount you have to spend on the mailing fee for a round trip to and from each factory.

In the example below: there are 8 factories (nodes 1 to 9) and you are at the location 0. The weights of edges indicate the mailing fees you have to pay for sending your computer.

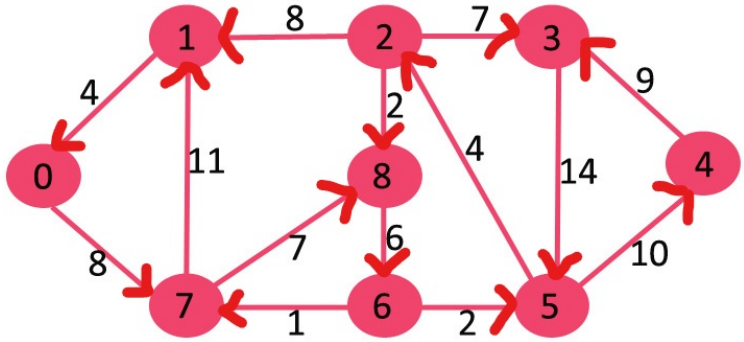


The answer for the above example is

factory	Mailing fees from the source
1	8
2	24
3	38
4	42
5	22
6	18
7	16
8	28

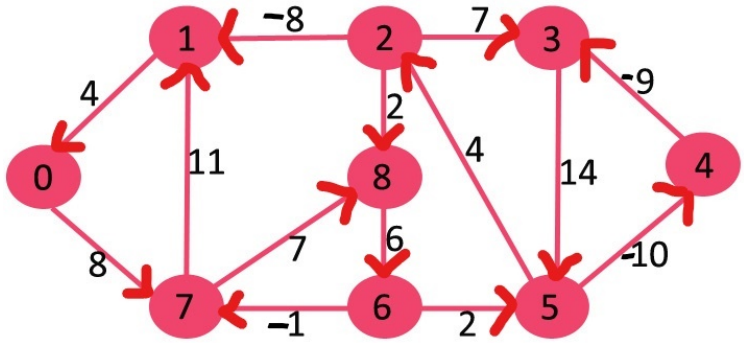
(1) (4pts) Please write down a pseudo code to solve the problem.

(2) (5pts) Due to some unexpected reasons, the paths to the factory change from bi-directional to uni-directional (as shown in the figure below). Please use the function you created above and explain how you can find the lowest mailing fee for a round trip to and from each factory. Your algorithm should run in $O(ElogE)$ time complexity. Briefly explain the correctness and why your algorithm meets the time complexity requirement.



Note: You do not have to write a pseudo code for this question.

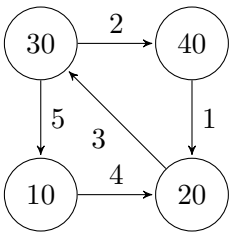
(3) (4pts) Now there are some offers provided by factories; some mailing fees change to negative, meaning that you can "earn" some money when using that path for sending your device. Please write a pseudo code to solve this problem. If there exists a cycle that you can actually "earn" money, please output "I am rich!"



Your algorithm should run in $O(N * E)$ time complexity (E is the number of edges). Briefly explain the correctness and why your algorithm meets the time complexity requirement.

(4) (3pts) Briefly compare these two algorithms you use in subproblem 2 and 3 (time and space complexity) and discuss their advantages and disadvantages.

(5) (6pts) Now each factory has a reliability value, we define a "trustful cycle" if the ratio of total reliability to total mailing fees is strictly larger than K ($K \in \mathbb{N}$). You can assume that there are no negative mailing fees for this subproblem. For example, in the example below, $K = 10$, the simple cycle route $30 \to 40 \to 20 \to 30$ has the total reliability $30 + 40 + 20 = 90$, and total mailing fees $2 + 3 + 1 = 6$. The ratio between them is 15, which is larger than K.



Please provide an algorithm to find if there exists a "trustful cycle". Your algorithm should run in $O(N * E)$ time complexity. Briefly explain the correctness and why your algorithm meets the time complexity requirement.

Hint: you can try reweighting the graph.

Problem F - Gaussian is Too Slow (Hand-Written) (8 points)

There are N unknown variables, namely $x_1...x_n$. Your task is to solve all variables by picking N equations from the "equation pool" below and make sure your solution of x is non-singular.
The following describes the equations' characteristics inside the equation pool:

- all coefficients for the unknown variables are 1. (ex: $x_1 + x_2$ is in the equation pool but $x_1 + 2x_2$ is not.)
- all variables inside an equation are consecutive. (ex: $x_2 + x_3 + x_4$ and x_3 are in the equation pool but $x_2 + x_4$ and $x_6 + x_1$ are not).
- each equation has their own "cost". (ex: equation $x_1 + x_2$ cost is 6, $x_2 + x_3$ is 3)
- you can assume all functions you choose will have a consistent solution for x.

Your task is to solve all variables x by picking N equations so that the total cost can be minimized.

As an example $N = 3$ and the "equation pool":

equation	cost
x_1	1
$x_1 + x_2$	5
$x_1 + x_2 + x_3$	2
x_2	6
$x_2 + x_3$	3
x_3	4

The solution for this example is to pick the equations x_1, $x_1 + x_2 + x_3$ and x_3. Thus, the total cost will be $1 + 2 + 4 = 7$.

Please reduce this problem to any algorithm you learned before (for example: Dijkstra, Bellman-Ford, Warshall, Kruskal, DP,...etc). Your algorithm should run in $O(N^2)$ time complexity. Briefly explain the correctness and why your algorithm meets the time complexity requirement.

Hint: you might need to add an extra node for this question.

Note: You do not need to write a pseudo code for this question. You just have to describe how you reduce this problem to any algorithm.