Data Structure and Algorithm

Homework #4

Due: 1:20pm, Tuesday, June 13, 2017

TA email: dsa1@csie.ntu.edu.tw

=== Homework submission instructions ===

- For Problem 1-3, please put all your solutions in a PDF file and name it *[StudentID]_hw4.pdf*, *e.g.*, *b05902987_hw4.pdf*. Please also **include your name and student ID on the first page** of the PDF file before submitting it to the online judge (http://140.112.91.212/judge/). You can either type them or write on papers and scan them. If you choose the latter, please make sure that your writing is recognizable and clear enough, otherwise you might receive some penalties. For Problem 4 and 5, they should be submitted individually and will be judged by the online judge (http://140.112.91.212/judge/).

- Discussions with others are encouraged. However, you should write down the solutions in your own words. In addition, for each problem you have to specify the references (the Internet URL you consulted and/or the people you discussed with) on the first page of your solution to that problem.

- For all programming problems, only C99 standard C language programs are accepted. (i.e., C++, or anything else, is not supported by the online judge system)

- For all the problems, up to one day of delay is allowed; however, you will get some penalties according to the following rule (the time will be in seconds):

$$\text{LATE SCORE} = \text{ORIGINAL SCORE} \times (1 - \frac{\text{Delay Time}}{86400})$$

  Penalty will be calculated separately for each of the following three parts (1) problems 1-3 (non-programming problems); (2) problem 4; (3) problem 5.

Problem 1. More on trees (20%)

In this problem, N is the number of nodes in the tree.

In problem 1.1.(b) and 1.2.(b), we highly recommend you to write down the pseudo code as well as explanation for your algorithm. If only one of them is given, please make sure it's written clearly enough. Otherwise, there might be penalty on your scores.

1. Centroid (7%)

   We define the *weight* of a tree to be the number of nodes in the tree. For example, the weight of the tree in Fig.1 is 8 and the weight of the tree in Fig.2 is 6. The weights of subtrees rooted at 1, 2, 3 in Fig.1 are 8, 3, 4, respectively.

   If we delete a node in a tree, the tree will split into a forest. For any such forest, we call the largest weight of all trees to be the *maximum tree weight* of this forest. A tree node is called the *centroid* if its deletion results in a forest with the *smallest* maximum tree weight. For example, the centroids of the tree in Fig.1 are node 1 and 3. (The maximum tree weight after deleting either of them is 4.) Note that there can be multiple centroids in a tree.

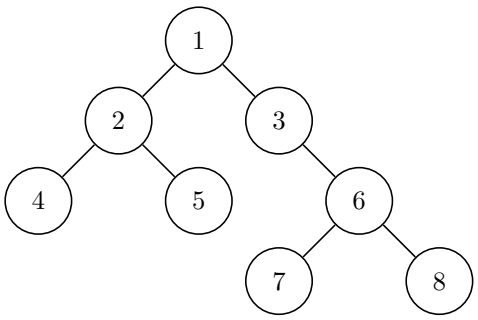


Fig. 1

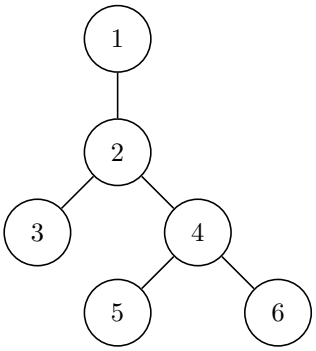


Fig. 2

   To find a centroid, you can arbitrarily pick a node as the root of the tree. Then DFS can be applied to determine the weights of all possible subtrees (with each node as the root of the subtree).

   (a) (3%) Give an algorithm to calculate all subtrees' weights in $O(N)$-time, where N is the number of nodes in the tree. Please give a pseudo code with simple explanation. For example, in Fig.1, the results of subtree weight at node $1, 2, \ldots, 8$ are 8, 3, 4, 1, 1, 3, 1 and 1 respectively.

   (b) (4%) Based on the solution in (a), please give an algorithm for finding one centroid for a given tree in $O(N)$ time.

2. Diameter (7%)

   This problem is about the diameter of a tree. A *diameter path* is any of the longest paths in the tree, and the *diameter* is the length (number of nodes - 1) of the diameter path. For example, the diameters of the trees in Fig.1 and Fig.2 are 5 and 3 respectively. Note that there can be multiple diameter paths in a tree, but only one diameter value.

   To find the diameter, you can arbitrarily pick a node as the root of the tree. Subsequently, use DFS to determine the heights of all possible subtrees (with each node as the root of the

subtree). The height of a tree is the maximum distance from root to a leaf.

(a) (3%) Calculate every subtree's height in O(N) time, and give the pseudo code with simple explanation. For example, in Fig.1, the results of subtree heights at node $1, 2, \ldots, 8$ are 4, 2, 3, 1, 1, 2, 1 and 1 respectively.

(b) (4%) Using the solution in (a), give an algorithm for finding the diameter of a given tree. Any solution similar to the algorithm in the following subproblem 1.3 is not allowed.

3. Midpoint (6%)

A *midpoint* of a tree is the node which has the smallest maximum distance to any other nodes in that tree. Note that there can be multiple midpoints in a tree. You can show that any midpoint is in the middle of some diameter path. (Specifically, if the diameter path contains an even number of nodes, then both of the middle two nodes are midpoints)

To find midpoints easily, we introduce an alternative algorithm for finding a diameter path of a given tree. Please read the pseudo code below.

```
1  int distance[N+100]; //N is number of nodes
2  int previous[N+100]; //It's a hint for (b)
3  void DFS(int u, int father, int dis):
4      // section A, add your code here
5      distance[u] = dis;
6      for each edge connected (u, v) :
7          if v == father:
8              continue
9          DFS(v, u, dis+1);
10
11 int get_farthest():
12     node = 1
13     for i = 2 to N:
14         if distance[i] > distance[node]:
15             node = i
16     return node
17
18 int solve():
19     root = 1;
20     DFS(root, -1, 0);
21     a = get_farthest();
22     DFS(a, -1, 0);
23     b = get_farthest(); // distance[b] is the diameter
24     // section B, add your code here
```

(a) (3%) In the function solve(), we picked the node 1 to be the root. Show that if we arbitrarily pick a node as the root, the farthest node from the root is always on a diameter path.

(b) (3%) Use the result of (a) that the length of path $(a \to b)$ is the diameter, please add the pseudo code in section A and B for finding a midpoint of a given tree. (Hint: you can use the previous array to record some information while running DFS.)

Problem 2. More Sorting (25% + 3%)

1. (6%)

   (a) (3%) Give an example that the running time of *QuickSort* (textbook 7.1) is $O(n^2)$.

   (b) (3%) Give an example that *InsertionSort* (textbook 2.1) can run faster than *MergeSort* (textbook 2.3.1).

2. (5%) Given n integers in the range 0 to k, please design an algorithm to preprocess the input in $\Theta(n + k)$ time and answer any query about how many integers fall into the range [a,b] in $O(1)$ time. Write down your algorithm in pseudo code and briefly show that the time complexity of any query is $O(1)$ and the preprocessing time is $\Theta(n + k)$.

3. (8%)

   (a) (3%) Use *RadixSort* (textbook 8.3) to sort the list (501, 939, 1137, 2345, 666, 34, 218). Set the base $r = 10$ and use *CountingSort* (textbook 8.2) to sort the base elements of each pass of *RadixSort*. Write down the sequence of the list at the end of each pass of *RadixSort*.

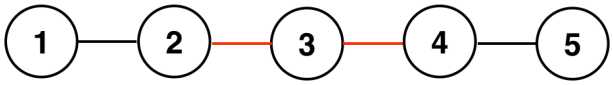
   (b) (5%) Use n, k and r to analyze the time complexities of *RadixSort* and the case in which we only use *CountingSort*. Then compare the computational cost between these two algorithms for sorting the list (501, 939, 1137, 2345, 666, 34, 218).

4. (6%) Given the sequence $[20, 29, 57, 37, 36, 50, 59]$, what's the sorting result of the modified *LSD RadixSort* using *MergeSort* (textbook 2.3.1) for each digit? And what's the sorting result of the modified *LSD RadixSort* using *HeapSort* (textbook 6.4) for each digit? Would the results of these 2 modified *LSD RadixSort* be the same? Why or why not?

5. (3%) Bonus
   Link : https://www.youtube.com/watch?v=U4lelYusKyQfeature=youtu.be
   In this video, TAs were sorting the numbers used in the class activity. Can you figure out what kind of sorting they were using? You only need to write down the name of the sorting algorithm.

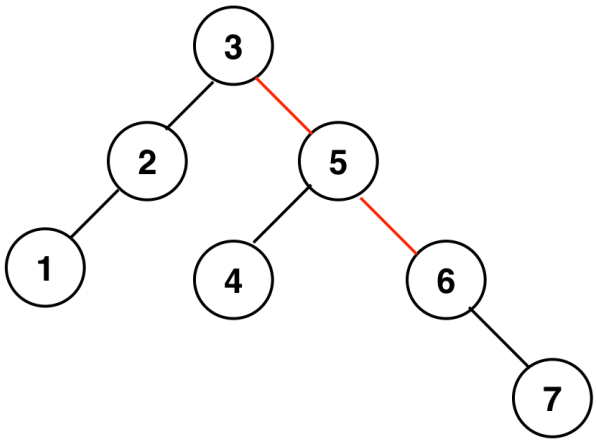
Problem 3. Disjoint Set (15%)

Given a tree T with N nodes and N-1 edges, which are either black(b) or red(r). Please find out how many tuples(a, b, c) of nodes exist, such that, all three paths, from a to b, from b to c, and from c to a, contain at least one red edge. Note that different permutations of three nodes, such as (a, b, c) and (b, c, a), are considered the same. Following is a tree with 5 nodes and 4 edges.



The valid tuples are (2, 3, 4), (2, 3, 5), (1, 3, 4) and (1, 3, 5). (2, 3, 4) is valid because from 2 to 3, from 3 to 4, and from 4 to 2, there are at least one red edge on the way. (1, 4, 5) is not a valid tuple since there is no red edges from 4 to 5.

1. (5%) Below is a tree with 7 nodes and 6 edges. Please list out all valid tuples, each with three nodes in ascending order.



2. (10%) Please design an algorithm to calculate the amount of valid tuples for a given input tree. Assume that N is the amount of nodes, and a set $E = \{E_1, E_2, ..., E_{N-1}\}$, where E_i= ($nodeA$, $nodeB$, $color$), indicating the color of the edge between node A and node B. Your algorithm should take N and E as input and return the number of valid tuples. Take the picture above for example. $N = 7$, and $E = \{(1, 2, b), (2, 3, b), (3, 5, r), (4, 5, b), (5, 6, r), (6, 7, b)\}$. The return value should be an integer equals to the amount of tuples you list in (3.1). Please explain your method in pseudo code and make sure that all details are dealt carefully. You are allowed to use as combination formula.

Problem 4. Tiger, Chicken, Worm, Bat, Go! (Programming problem) (20%)

Eddy likes to play a game called "Tiger, Chicken, Worm, Bat, Go!". It's a game similar to "Paper, Scissor, Stone". Two players involved in this game each choose one of "Tiger", "Chicken", "Worm", or "Bat". After that, the two players speak out "Tiger, Chicken, Worm, Bat Go!" loudly and show their choices. Let (X,Y) represent that player one chooses X and player two chooses Y. Then, one of following outcome will happen to player one:

- `Win`: If one of ("Tiger", "Chicken"), ("Chicken", "Worm"), ("Worm", "Bat"), ("Bat", "Tiger") happens. *(Since tiger can bite chicken, chicken can eat worm, worm can decay bat, and bat can beat tiger)*
- `Lose`: If one of ("Tiger", "Bat"), ("Chicken", "Tiger"), ("Worm", "Chicken"), ("Bat", "Worm") happens.
- `Same`: If one of ("Tiger", "Tiger"), ("Chicken", "Chicken"), ("Worm", "Worm"), ("Bat", "Bat") happens.
- `Nothing`: If one of ("Tiger", "Worm"), ("Chicken", "Bat"), ("Worm", "Tiger"), ("Bat", "Chicken") happens.

However, Eddy couldn't find any friends willing to play this game, and decided to make some robots to play with him. Since building an artificial intelligence system is too hard for Eddy, each of his robots exhibits a fixed pattern of always choosing one of those four things. (For example, 1-st robot will always choose "Tiger", 2-nd robot will always choose "Bat", and so on.)

After finishing making all of the robots, Eddy wants to test them. For each test, Eddy will choose two robots, one as player one and the other as player two, and record the result of outcome of player one. However, recording anything is still too hard for Eddy. Thus, Eddy asks you to help him record the result. He will tell the result one by one. But, some of the result may be **invalid** since Eddy isn't so clever. You need to keep track of only **valid** results.

- A result is **invalid** if it conflicts with any of the results recorded so far. The invalid result is then discarded and not recorded.
- Otherwise, the result is **valid** and you need to record it.

To make sure that you are actually helping Eddy record the **valid** result, Eddy may sometimes ask you what's the outcome when `x`-th robot plays as player one and `y`-th robot plays as player two. You need to answer based on the **valid** results you record so far. If based on recorded **valid** results so far, you can't determine the outcome, you should also report it to Eddy.

Input Format

The first line contains T indicating there are T independent scenarios.

For each scenario: first line contains two positive integer N, Q indicating that Eddy makes N robots and he will tell or ask you an outcome for a total of Q times.

For following Q lines, each line consists one of following format:

- `R x y C`: Eddy tells you the result when `x`-th robot plays as player one and `y`-th robot plays as player two, the outcome of player one is `C`. `C` will be one of 'W'(Win), 'L'(Lose), 'S'(Same), 'N'(Nothing).
- `A x y`: Eddy asks you what's the outcome of player one when `x`-th robot plays as player one and `y`-th robot plays as player two.

Input Constraint

It is guaranteed that

- $N \leq 10^6$
- $Q \leq 10^6$
- In each scenario, $1 \leq$ `x` $\neq$ `y` $\leq N$.
- `C` is one of 'W', 'L', 'S', or 'N'.
- Sum of N over T scenarios is less than or equal to 10^6.(*i.e.*, $\sum N \leq 10^6$)
- Sum of Q over T scenarios is less than or equal to 10^6.(*i.e.*, $\sum Q \leq 10^6$)

Output format

For each scenario, you should output Q lines.

For `R x y C`, you should output `'O'` (15-th uppercase English letter) without quotation mark in one line if it's **valid** otherwise you should output `'X'` without quotation mark in one line.

For `A x y`, you shuold output `'W'` without quotation mark in one line if the outcome of player one is `Win`, output `'L'` without quotation if the outcome is `Lose`, output `'S'` without quotation if the outcome is `Same`, output `'N'` without quotation if the outcome is `Nothing`, output `'?'` without quotation if you can't determine the outcome based on the currently recorded results.

Sample Input

```
2
4 9
R 1 2 W
R 2 3 W
R 3 4 W
A 1 2
A 1 3
A 1 4
A 2 3
A 2 4
A 3 4
4 5
R 1 2 W
R 2 3 W
R 3 1 W
A 1 4
A 3 1
```

Sample Output

```
O
O
O
W
N
L
W
N
W
O
O
X
?
N
```

Hint

1. For first scenario of sample input:
   First three results are all **valid**. Let $[A, B, C, D]$ represent 1st robot chooses A, 2nd robot chooses B, and so on. Then, one of following combination must happen: ["Tiger", "Chicken", "Worm", "Bat"], ["Chicken", "Worm", "Bat", "Tiger"], ["Worm", "Bat", "Tiger","Chicken"], ["Bat", "Tiger", "Chicken", "Worm"]. No matter which combination happens, the outcome of each pair of robots can be determined and is unique.

2. For second scenario of sample input:
   The first two results are **valid**. From the first two results, for the first three robots, one of following combination must happen: ["Tiger", "Chicken", "Worm"], ["Chicken", "Worm", "Bat"], ["Worm", "Bat", "Tiger"], ["Bat", "Tiger", "Chicken"]. No matter which one happens, the third robot won't be able to win over the first robot. Thus, the third result is **invalid**. Since we have no idea about the 4-th robot, we can't determine the result of the first robot versus the 4-th one yet. For the last request, no matter which combination mentioned above happens, the outcome for third robot will be 'N'(Nothing).

Problem 5. Portal (Programming problem) (20%)

In the kingdom of Difficult Subject Again, a.k.a. DSA kingdom, there are n cities, numbered through 1 to n.

There are m portals, the i-th portal can transport a person from city a_i to city b_i. (Note: The portal is not bidirectional, *i.e.*, the i-th portal does *not* transport a person from city b_i to city a_i.)

DSA kingdom is going to hold International Computing Power Contest, a.k.a. ICPC. They have to transport a lot of guests from city 1, where the airport stands, to city n, in which the event will be held. To treat the guests nicely, the government will only be transporting the guests with portals among the cities. In addition, the government will find a shortest sequence of portals for the guests to arrive city n since it is more efficient to use fewer portals for transpotation. Let's call the length of such sequence as the cost of transportation.

Now, the government would like to build another portal for the ICPC event. Because of technical issues, there are only q possible choices for the new-built portal, the i-th of the q choices can transport a person from city c_i to city d_i. Again, but not city d_i to city c_i. The government needs your help to decide which one is the best choice. For each choice i, you have to measure the cost of transportation if the new portal is built there.

Input Format

The first line contains a positive integer, T, the number of test cases, and T test case(s) follow.

For each test case, the first line contains three positive integers, n, m and q, indicating the number of cities, the number of existing portals and the number of possible choices to build a new portal. The following m lines each containing two positive integers, the two integers in i-th line are a_i and b_i, indicating the two end points of the i-th portal. Then, q lines follow, each containing two positive integers, the two integers in i-th line are c_i and d_i, indicating the two end points of the i-th choice for the new-built portal.

Input Constraint

It is guaranteed that

- $n \geq 2$
- Sum of n over T test cases is less than or equal to 2×10^5.
- Sum of m over T test cases is less than or equal to 5×10^5.
- Sum of q over T test cases is less than or equal to 10^5.
- For 40% points, the sum of n, the sum of m and the sum of q over T test cases are all less than or equal to 10^3.
- $1 \leq a_i, b_i, c_i, d_i \leq n$
- there exists a way to transport guests from city 1 to city n using only the m existing portals.

Output format

For each test case, you should output q lines.

The i-th line of a test case should contain the cost of transportation if we choose the i-th of the total q choices to build the new portal.

Sample Input

```
2
8 9 5
1 2
1 3
2 4
2 6
3 5
4 7
5 6
6 8
7 8
6 1
1 6
1 2
5 8
2 8
6 5 3
1 2
2 3
3 4
4 5
5 6
2 5
1 6
3 2
```

Sample Output

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