Homework #3

Red Correction Date: 11/15/2022 14:30

Due Time: 2022/12/01 14:20

Contact TAs: `ada-ta@csie.ntu.edu.tw`

Instructions and Announcements

- There are **four programming problems** and **two 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 should upload your answer to **Gradescope** as demonstrated in class. For each sub-problem, please label (on Gradescope) the corresponding pages where your work shows up. 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 of your solution to that problem. You may get zero point due to the lack of references.
- **Tips for programming problems.** Since the input files for some programming problems may be large, please add
  - `std::ios_base::sync_with_stdio(false);`
  - `std::cin.tie(nullptr);`

  to the beginning of the main function if you are using **`std::cin`**.

Problem 1 - Amortized Hanoi (Programming) (10 points)

Problem Description

AA, BB, CC, and DD like to play the Tower of Hanoi.

Today, they are playing a similar game with only one rod and with dangos instead of disks. Each dango has a weight. The similarity resides in that, given a rod with many dangos, one can only take the topmost dango off the rod or put a dango on the top (i.e. the structure of the dangos on the rod is like a stack).

Each person has a preference. AA likes to put a dango on the rod. BB likes to eat one dango. CC likes to eat heavy dangos and calculates the total weight of the dangos eaten. DD dislikes eating dango but likes to calculate the sum of the weight of dangos.

The game lasts for M rounds, each consisting of an operation—`PUT`, `TAKE`, `CC`, or `DD`—defined below.

Input

The first line of input contains an integer M, indicating the number of rounds. Each of the following M lines indicates what happens in the i-th round.

Each line contains a capitalized string and some (possibly zero) integers. Each of them is one of the followings.

1. `PUT w`: AA puts a dango with weight w to the rod. After this operation, you don't need to output anything.
2. `TAKE`: BB takes a dango off the rod and eats it. Please output the weight of the dango BB eats after this operation.
3. `CC x k`: CC takes x dangos off the rod, eats the $\lceil \frac{x}{k} \rceil$ heaviest ones, and puts the remaining ones back in the original order. After this operation, please output the sum of the weight of dangos that CC eats this time.
4. `DD x`: DD takes x dangos off the rod, calculates the sum of the weight of those dangos, and puts them back in the original order. After this operation, please output the sum DD gets.

Here, "the original order" means if $\Big($(dango A is below dango B before the operation) and $\big($both A, B are to put back to the rod afterwards$\big)\Big)$, then A should still be placed below B after the operation.

- $1 \leq M \leq 10^6$
- In `PUT` operations, $1 \leq w \leq 10^9$. It is guaranteed that among all `PUT` operations, w are distinct.
- For operation `TAKE`, it is guaranteed that $S_i \geq 1$ where S_i is the number of dangos on the rod right before the i-th operation.
- For operation `CC`, `DD`, it is guaranteed that $1 \leq x \leq S_i$ where S_i is the number of dangos on the rod right before the i-th operation.
- In operation `CC`, $1 \leq k \leq 5$.

Output

The program should print to the standard output at most M lines containing the required result.

Subtask 1 (10 %)

- $M \leq 1000$.

Subtask 2 (20 %)

- There are no operation `DD`.

Subtask 3 (30 %)

- There are no operation `CC`.

Subtask 4 (40 %)

- No other constraints.

Sample Input 1

```
12
PUT 13
PUT 87
PUT 99
PUT 22
DD 3
PUT 43
PUT 37
CC 6 3
PUT 21
CC 1 1
PUT 73
DD 5
```

Sample Output 1

```
208
186
21
188
```

Sample Input 2

```
17
PUT 43
PUT 59
PUT 15
PUT 96
DD 2
DD 4
PUT 36
TAKE
PUT 22
DD 1
DD 3
PUT 88
CC 6 5
PUT 50
PUT 57
PUT 83
DD 2
```

Sample Output 2

```
111
213
36
22
133
184
140
```

Hint

- This problem can be solved without advanced data structures, e.g. any balanced binary search tree.

Problem 2 - Emo Driver (Programming) (10 points)

Problem Description

Thomas is emo. He hates everything. He doesn't want to work. He only wants to drive his beloved car, escape from the miserable city where he lost everything, and travel all around ADA kingdom.

Thomas lives in the capital of ADA kingdom with N cities and M roads, each road in ADA kingdom has its length, expressed in miles. All the roads are one-way roads; that is, if there is a road from city a to city b, you can not reach city a directly from city b unless there is another road from city b to city a. To vent his negative emotions, Thomas, for the i-th of the next $N-1$ days, will drive to the $(i+1)$-th city from the capital, fill himself with some alcohol in a bar there and then hire a driver to drive him back home. Since Thomas always has a clear mindset when driving to the bar, he drives along the optimal way with minimal distance.

ZCK, a good friend of Thomas, wants to help him with his teleportation magic. He gives Thomas a chance to build a one-way portal from city a to city b today. You can consider the portal a road from city a to city b with zero length. Though Thomas is emo, he is still rational. He will find the best way to build the portal to minimize the total miles he drives.

Unfortunately, Thomas is drunk now. Please help him determine the minimum total miles he needs to drive for the next $N-1$ days after optimally building the teleport.

It is guaranteed that Thomas can reach each city by some roads starting from the capital, but it is **not** guaranteed that each city can get back to the capital by the roads (the driver hired by Thomas has some superpower that we don't know).

Input

The first line contains two integers N and M, which are the number of cities and roads.

In the next following M lines, each contains three integers a, b, c, meaning that there is a road from city a to city b with c miles.

Note that the capital has index 1.

- $1 \leq N \leq 2000$
- $1 \leq M \leq \min\left(4000, \frac{N(N-1)}{2}\right)$
- $1 \leq a_i, b_i \leq N$
- $0 < c_i \leq 10^4$

Output

Print a single integer representing the minimum total miles he needs to drive for the next $N-1$ days after optimally building the teleport.

Test Group 0 (0 %)

- Sample input

Test Group 1 (50 %)

- $1 \leq N \leq 200$
- $1 \leq M \leq \min\left(1000, \frac{N(N-1)}{2}\right)$

Test Group 2 (50 %)

- No additional constraint

Sample Input 1

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

Sample Output 1

```
4
```

Sample Input 2

```
5 6
1 2 3
1 3 4
2 3 5
3 4 2
4 5 1
1 5 3
```

Sample Output 2

```
8
```

Sample Input 3

```
5 10
1 2 9
1 3 8
1 4 7
1 5 6
2 3 1
2 4 2
2 5 3
3 4 6
3 5 1
4 5 2
```

Sample Output 3

```
5
```

Notes

- In the first case, building the portal from 1 to 3 achieves the goal.
- In the second case, building the teleport from 1 to 3 achieves the goal.
- In the third case, building the teleport from 1 to 2 achieves the goal.

Problem 3 - Three-Mouthed Sheep and the Forest (Programming) (15 points)

Problem Description

Three-mouthed sheep lives in a forest, which is, by definition, an undirected graph in which any two vertices are connected by at most one path. Equivalently, a forest is an undirected acyclic graph, all of whose connected components are trees; in other words, the graph consists of a disjoint union of trees. The trees in this forest are rooted and all the vertices are initially white.

Three-mouthed sheep would like to make the forest greener. To achieve this goal, he can perform the two following operations.

- Spend c_i dollars to paint the i-th vertex green.
- Spend d_i dollars to paint all the vertices in the subtree rooted at the i-th vertex green, including the i-th vertex itself.

He defines the greenness of the forest as $\sum_{\text{vertex } i \text{ is green}} c_i$. With M dollars in hand, he wants to maximize the greenness by performing the operations optimally. Could you do him a favor?

Input

The first line contains two integers N and M, indicating the numbers of vertices in the forest and the money three-mouthed sheep has in dollars.

The second lines contains N integers $f_0, f_1, \ldots, f_{N-1}$, indicating the parents of the vertices. More precisely, vertex f_i is the parent vertex of vertex i if $f_i \neq -1$; otherwise, vertex i is one of the roots of the trees. Note that the indices are 0-based.

Then, N lines follows. Each of these lines contains two integers c_i and d_i. Please refer to the problem description for their meanings.

In other words, the input is presented in the following format:

$$
\begin{array}{ll}
N & M \\
f_0 & f_1 \quad \cdots \quad f_{N-1} \\
c_0 & d_0 \\
c_1 & d_1 \\
\vdots & \\
c_{N-1} & d_{N-1}
\end{array}
$$

- $1 \leq N, M \leq 3000$
- $-1 \leq f_i \leq i - 1$
- $1 \leq c_i, d_i \leq 3000$

Output

Output the maximum greenness if three-mouthed sheep paint the vertices optimally.

Test Group 0 (0 %)

- Sample input

Test Group 1 (30 %)

- $f_i = -1, \forall 1 \leq i \leq N$

Test Group 2 (30 %)

- $1 \leq N, M \leq 100$

Test Group 3 (40 %)

- No additional constraint

Sample Input 1

```
1 1
-1
1 1
```

Sample Output 1

```
1
```

Sample Input 2

```
10 10
-1 0 0 0 1 3 -1 -1 6 4
6 7
3 8
2 3
5 105
3 9
2 3
2 7
1 8
77 3
2 9
```

Sample Output 2

```
100
```

Notes

In sample 2, three-mouthed sheep paints the 8-th vertex with the first operation and the vertices in the subtree rooted at the 0-th vertex.

Problem 4 - Highways in Kingdom (Programming) (15 points)

Problem Description

Arctan is the president of ADA kingdom. There are n cities in the kingdom. For some pairs of the cities, there is a road connecting them. For each pair of cities, one can go from one city to another along some (possibly one) roads. When building a highway on the road r, the length of the highway is l_r and the toll is c_r. Arctan wants to build highways on exactly $n-1$ roads, so that all cities can be reached from the capital, which is the city 0, by highways. What's more, he needs to minimize the total length of highways.

When a person goes from a city to another by highways, the person will be charged on a highway if and only if that highway is one of the longest highways in their trip. In another word, let $C(u,v)$ denote the total toll that is charged to go from u to v by highways, and $P(u,v)$ denote the path from u to v, then

$$C(u,v) := \left(\sum_{r \text{ is one of the highways in } P(u,v) \text{ and it is one of the longest highways in } P(u,v)} c_r \right)$$

Arctan is a good president, and he opens a president hour to hear the voice of people every Tuesday. Because he wants to encourage people to go to his president hour, he wants to minimize the total cost to go from every city to the capital. In another word, he wants to minimize $\sum_{v=1}^{n-1} C(v,0)$.

Calculate the minimum possible value of $\sum_{v=1}^{n-1} C(v,0)$ under the constraint where the total length of all highways is minimized.

Input

The first line of input contains two integers n, m, the number of cities and the number of roads in ADA kingdom. Each of the following m lines has four space-separated integers u_i, v_i, l_i, c_i, indicating the i-th road is connecting to u_i, v_i, and the length is l_i, the toll is c_i.

- $2 \le n \le 4 \times 10^5$
- $n-1 \le m \le \min(4 \times 10^5, \frac{n(n-1)}{2})$
- $0 \le u_i, v_i \le n-1,\ u_i \ne v_i$
- $1 \le l_i, c_i \le 10^7$
- It is guaranteed that no two roads connects to the same pair of cities.

Output

Output an integer k denoting the minimum possible value of the total cost to go from every cities to the capital under the constraint the total length of all highways is minimized.

Test Group 0 (0 %)

- Sample input

Test Group 1 (10 %)

- $n, m \leq 3000$

Test Group 2 (20 %)

- $\forall i \neq j,\ l_i \neq l_j$

Test Group 3 (20 %)

- $\forall i,\ l_i = 1$

Test Group 4 (50 %)

- No other constraints

Sample Input 1

```
4 5
0 1 3 1
0 2 2 4
1 2 1 7
1 3 4 3
2 3 5 2
```

Sample Output 1

```
11
```

Sample Input 2

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

Sample Output 2

```
11
```

Sample Input 3

```
5 10
0 1 2 1
0 2 7 4
0 3 1 1
0 4 8 4
1 2 2 2
1 3 8 1
1 4 1 3
2 3 8 5
2 4 2 6
3 4 8 2
```

Sample Output 3

```
6
```

Notes

In the third sample, picking the edges $(0, 1), (0, 3), (1, 2), (1, 4)$ will be the answer.

Note that $C(1, 0) = 1, C(2, 0) = 1 + 2 = 3, C(3, 0) = 1$.

$C(4, 0) = 1$ because the path from 4 to 0 consists of $(4, 1), (1, 0)$, and $(4, 1)$ is not the longest highway in this path, the toll on $(4, 1)$ is not counted in $C(4, 0)$.

Problem 5 - Professor Cheng (Hand-Written) (30 points)

Note: In this problem, you are not allowed to write pseudocode. Please explain your algorithm in words. Some explanation about "why your algorithm is correct" is needed.

Part 1.

Professor Cheng offers a course – Algorithm Design and Orienteering (ADO) – this semester. Arctan is a student in Professor Cheng's ADO class. As he is not as smart as Professor Cheng, he asks you for some help. Please help him to solve these problems!

(a) (4 points)

Professor Cheng taught SCC (strongly connected component) in class. Arctan claims that

$$\text{Vertices } a \text{ and } b \text{ are in the same SCC} \Rightarrow \Big(f_a < f_c < f_b \Rightarrow \text{Vertices } a, b \text{ and } c \text{ are in the same SCC}\Big),$$

where f_v denotes the finishing time of vertex v when performing DFS on the graph.

Please give a proof of this statement if Arctan is correct or a counterexample if he's wrong. In the former case, please write your proof **concisely**. In the latter case, your counterexample should contain **no more than 10 edges** and you should briefly explain your counterexample.

(b) (4 points)

The homework for ADO is to travel to the farthest MRT station from where you live. However, Arctan is a lazy student; he wants to travel as less as he can. Find out an MRT station (in Taipei Metro) from which taking a ride to the farthest station from it costs the least. The cost of a MRT station is the most expensive travel fare from this station. Briefly explain (in less than five sentences) how you find the answer.

Note: In this problem you can install the App "Go! Taipei Metro" (or "台北捷運 Go" if you prefer Chinese to English) to find out the travel fare from any station to one another. Alternatively, you can check out this website. You only need to consider the route map formed by the following six lines: Tamsui-Xinyi line, Songshan-Xindian line, Bannan line, Zonghe-Xinlu line, Wenhu line, and Circular line.

Part 2.

In the department of CSIE, there are N Professor Chengs labeled from 1 to N. If you ask one of the Professors Chengs who Professor Cheng is, he/she will point to exactly one Professor Cheng (possibly himself/herself) and say he/she is Professor Cheng. That is, in this professor-pointing graph, every vertex has exactly one out-degree. Let $F(n)$ be the Professor Cheng's index that Professor Cheng n is pointing at and $F_k(n) = \underbrace{F(F(\cdots F(n)\cdots))}_{k\ F\text{'s}}$.

In this part, you can obtain the points from both subproblems (d) and (e) by only writing a solution to subproblem (e). If this is your case, please label the same pages for subproblem (d) and (e) on Gradescope for that piece of solution.

(c) (4 points)

Prove that for any n, there exists a pointing-cycle in the sequence $a_i = F_i(n)$. That is, there exists some $\aleph_0, T > 0$ such that for all $\aleph > \aleph_0, a_\aleph = a_{\aleph+T}$.

(d) (4 points)

Find out the index of Professor Cheng after K steps of pointing from Professor Cheng 1. That is, find out $F_K(1) = \underbrace{F(F(\cdots F(1)\cdots))}_{K\ F\text{'s}}$. Your algorithm should run in $O(N)$ time which should be independent of K. You may assume that the modulo operation can be done in $O(1)$ time.

(e) (4 points)

Find out the indices of Professor Cheng after K steps of pointing starting from every Professor Cheng. That is, find out N numbers, $F_K(i) = \underbrace{F(F(\cdots F(i)\cdots))}_{K\ F\text{'s}}, \forall i = 1, \ldots, N$. Your algorithm should run in $O(N)$ time which should be independent of K. You may assume that the modulo operation can be done in $O(1)$ time.

Part 3.

Now the N Professor Chengs can point to any number of Professor Chengs (possibly zero). A "professor-pointing chain" is a sequence of Professor Chengs where every Professor Cheng except the last one is pointing at the next Professor Cheng in this sequence. Any Professor Cheng can appear multiple times in this sequence. Balute, as a Professor Cheng researcher, wants to know how long the "professor-pointing chain" can be.

In this part, you can obtain the points from both subproblems (f) and (g) by only writing a solution to subproblem (g). If this is your case, please label the same pages for subproblem (f) and (g) on Gradescope for that piece of solution.

(f) (6 points)

If the professor-pointing graph forms a DAG (directed acyclic graph), please help Balute to find the length of the longest professor-pointing chain starting from each vertex in $O(V + E)$ time, where V is the number of Professor Chengs (i.e., N) and E is the number of total edges (i.e., the summation of the number of Professor Chengs each Professor Cheng points at) in the professor-pointing graph.

(g) (4 points)

For any general professor-pointing graph, please help Balute to find the longest professor-pointing chain starting from each vertex in $O(V + E)$ time, where V is the number of Professor Chengs (i.e., N) and E is the number of total edges (i.e., the summation of the number of Professor Chengs each Professor Cheng points at) in the professor-pointing graph. If there exists a Professor Cheng i such that a professor-pointing graph originating from him/her can be arbitrarily long, you should mark the answer for Professor Cheng i as ∞.

Problem 6 - ADA Guild (Hand-Written) (20 points)

Note: In this problem, you are not allowed to write pseudocode. Please explain your algorithm in words. Also note that you should prove the time complexity using the method specified in each problem statement.

The ADA Guild is the most prestigious guild in the ADA Kingdom. As a well-known S-Rank adventurer in the kingdom, you are welcomed by the ADA Guild to join them. However, you still have to obtain the approval from the Sage, who would like to test your intelligence.

The Sage notices that you have solved Problem 1 in ADA HW3. How amazing! He asks if you could also explain how you have solved it.

(a) (5 points)

Briefly describe the algorithm you used in Problem 1, and prove the amortized time complexity using the **accounting method**. It is required that your total time complexity is $o\left(kM^{1.1101420}\right)$. In this problem, you are not allowed to use any binary search tree(BST) to solve this problem. However, if you use binary search tree to solve Problem 1, you can still get the points from Problem 1.

Knowing that you are attending ADA, which is the best course in the kingdom, the Sage asks you to recall the *dynamic table* from the lecture on amortized analysis.

(b) (5 points)

Consider a variant of the *insert-only dynamic table* as defined in the lecture. When out of slots, instead of doubling the size of the table, here we let the new table have the size of the next Fibonacci number. Fibonacci numbers are numbers in the Fibonacci sequence, which is defined by the recurrence relation $F_0 = F_1 = 1$ and $F_n = F_{n-1} + F_{n-2}$ for $n > 1$. Give the tightest amortized time complexity per operation for a sequence of N insertions, and prove it using the **potential method**.

(c) (5 points)

Consider the following problem: A line L is divided into C cells, denoted by an array $L[1..C]$. Initially, all cells have the color 0. Then N operations are performed, where the i-th operation is given by (l_i, r_i, c_i), indicating that all the cells in the segment $L[l_i..r_i]$ are colored with the color c_i.

After performing any number of operations, we can divide the line into segments where all the cells in a segment share the same color. You should maintain a data structure which stores these segments in order, and a tuple (l, r, c) is stored for each segment, where l and r are the indices of the leftmost and rightmost cells in the segment and c is the color of the segment. As an example, consider performing the sequence of operations $(2, 6, 1), (4, 5, 2)$ and $C = 10^9$, then the data structure contains $(1, 1, 0), (2, 3, 1), (4, 5, 2), (6, 6, 1), (7, 10^9, 0)$.

Design an algorithm that maintains a **balanced binary search tree** as an implementation of this data structure with an amortized time complexity of $O(\log N)$ per operation. You don't need to prove the time complexity, but you can briefly justify it within two sentences. As a consequence, you should then be able to obtain the color of a given cell in $O(\log N)$ time.

Note that a balanced binary search tree supports insertion, deletion, finding the next/previous node, and finding the first node with key greater than (or equal to) a given value, each in $O(\log N)$ time.

Congratulations! You have joined the ADA Guild.

Upon entering, you meet Acein, someone who often writes *fake solutions* when solving problems. He tells you that the Sage asked him to solve a problem, and he provided an algorithm which he thought was naive and thus too inefficient, but was somehow accepted by the Sage. He thinks it was again a fake solution. But could it be that he is just unfamiliar with amortized analysis?

(d) (5 points)

Consider the following problem: Given a root tree $T = (V, E)$ where each vertex $v \in V$ has a weight $v.w$. For each $v \in V$, let $child(v)$ consist of all the children of v and $subtree(v)$ consist of all the vertices in the subtree rooted at v, and for simplicity, assume they are already stored in corresponding arrays so the tree traversal procedure is omitted in the pseudocode.

We wish to compute

$$\sum_{u \in child(v)} \sum_{w \in child(v) \setminus \{u\}} \sum_{x \in subtree(u)} \sum_{y \in subtree(w)} (x.w \times y.w)$$

for each $v \in V$. That is, in one selection we select two distinct children u, w of v, then select a vertex x from the subtree rooted at u and a vertex y from the subtree rooted at w, and we sum up the values of $x.w \times y.w$ from all the possible selections.

The following provides an algorithm for this problem.

Algorithm 1: COMPUTE(T, v)

```
sum = 0
for each u ∈ child(v) do
    for each w ∈ child(v)\{u} do
        for each x ∈ subtree(u) do
            for each y ∈ subtree(w) do
                sum = sum + x.w × y.w
            end
        end
    end
end
return sum
```

Prove that after computing for all vertices $v \in V$, the amortized time complexity is $O(V)$ for each vertex using the **aggregate method**. And hence, the total time complexity after running Algorithm 1 on all vertices is $O(V^2)$.