CodeWOOT (3978)

AoPS Staff

Thursday
Sep 5, 2024 - Mar 6, 2025
7:30 - 9:30 PM ET (4:30 - 6:30 PM PT)

Homework

Homework: Lesson 2

You have **8** writing problems to complete.
Due **Sep 25**.

Readings

Lesson 2 Transcript: Thu, Sep 12 - A
Lesson 2 Transcript: Thu, Sep 12 - B

Challenge Problems

Total Score: 0

Problem 1 (46481)

Problem 2-1: From Everywhere To Everywhere: Report Error

Farmer John has N farms ($2 \leq N \leq 750$) numbered 1 through N, and connected by M one-directional roads ($1 \leq M \leq 3 \cdot 10^5$). The i-th of these roads connects farms A_i and B_i $(1 \leq A_i, B_i \leq N)$, and takes C_i units of energy to use $\left(-10^9 \leq C_i \leq 10^9\right)$. Notice that the cost can be negative -- perhaps a road is so beautiful that traveling along it is energizing rather than exhausting!

For any ordered pair of farms, there is at most one road directly connecting the first to the second. It is guaranteed that each farm is reachable from each other farm via some sequence of roads.

Bessie has gotten a bunch of her friends together. Each friend is given an ordered pair of two different farms: a start and a goal. For any possible combination of start and goal, exactly one friend has that combination.

Each friend will take the path from their start farm to their goal farm that minimizes the total cost of their path (the sum of all the C_i values of the roads in the path). It is guaranteed that there is no way for a cow to get an arbitrarily negative total cost.

Determine the sum of these total costs, for all cows.

INPUT FORMAT (input arrives from the terminal / stdin):

The first line contains N and M.
Each of the next M lines contains A_i, B_i, and C_i.

OUTPUT FORMAT (print output to the terminal / stdout):

Print one line with the sum of the total costs for all cows.

SAMPLE INPUT:

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

```
2 3 2
3 2 6
```

SAMPLE OUTPUT:

```
36
```

SCORING:

In test cases 2-5, $N \leq 50$.
In test cases 6-9, $N \leq 250$.
In test cases 4-7, all $C_i \geq 0$.
Test cases 10-20 satisfy no additional constraints.

Languages: C++17

Source File:

Problem 2 (46463)

Problem 2-2: Trust Circles: Report Error

Consider a social network graph of N people ($1 \leq N \leq 10^5$), numbered 1 through N. The graph has M directed edges ($N \leq M \leq 2 \cdot 10^5$), each of which goes from one person to another. It is guaranteed that at each person is the origin of at least one directed edge.

We say that a person A trusts another person B if there is a directed edge from A to B in the network, or if there is a directed edge from A to some other person P_1 and a directed edge from P_1 to B, and so on.

A trust circle is a group (of at least one person) such that every person in the group trusts every other person. A **maximal trust circle** is a trust circle that is not part of any larger trust circle. Notice that it is possible for a trust circle to consist of just one person -- since there is no "every other person" to worry about in that case, the definition trivially holds.

Determine the total number of maximal trust circles in the network.

INPUT FORMAT (input arrives from the terminal / stdin):

The first line contains N and M.

Each of the next M lines is of the form A_i, B_i, indicating that there is a directed edge from person A_i to person B_i. No line is repeated.

OUTPUT FORMAT (print output to the terminal / stdout):

Print the total number of trust circles in the network.

SAMPLE INPUT:

```
3 3
3 1
```

```
2 1
1 2
```

SAMPLE OUTPUT:

```
2
```

SCORING:

In test cases 2-6, each person has exactly one outgoing directed edge.
Test cases 5-9 satisfy $N \leq 10^4$.
Test cases 10-20 satisfy no additional constraints.

Languages: C++17

Source File:

Problem 3 (46306)

Problem 2-3: Billionaire: Report Error

You are in an antiques marketplace that has S stalls ($1 \leq S \leq 10000$) and deals in up to N different types of items ($2 \leq N \leq 2000$). You currently have 1000000 (10^6) dollars and one copy of item 1, and nothing else.

Each stall trades one specified item for another one item, with some money also possibly changing hands. Specifically, at stall i, you can trade item A_i plus D_i dollars (where $-1000 \leq D_i \leq 1000$) for item B_i ($1 \leq A_i \leq N, 1 \leq B_i \leq N$). If D_i is positive, you cannot trade at that stall unless you have at least D_i dollars. If D_i is negative, instead of having to pay, you will *receive* $|D_i|$ dollars as part of the trade.

You may use (and even re-use) each stall as many times as you want, and you do not necessarily have to use every stall.

Determine whether it is possible for you to have at least X dollars ($10^6 < X \leq 10^9$) at any point.

INPUT FORMAT (input arrives from the terminal / stdin):

Each test case begins with an integer T ($1 \leq T \leq 20$): the number of subproblems in the test case. Then, each subproblem consists of:
* one line with S, N, and X.
* S more lines; the i-th of these represents the i-th stall, and has the values A_i, B_i, D_i.

OUTPUT FORMAT (print output to the terminal / stdout):

For each subproblem in the case, print one line with either $\texttt{YES}$ or $\texttt{NO}$.

SAMPLE INPUT:

```
2
2 3 1000001
1 2 100
2 3 -200
```

```
6 5 1000001
3 5 -100
1 3 200
2 3 500
5 2 50
1 3 600
2 1 -100
```

SAMPLE OUTPUT:

```
YES
NO
```

SCORING:

Test cases 2-5 satisfy $N \leq 200$.
Test cases 2-3 and 6-7 satisfy $X = 10^6 + 1$.
Test cases 7-9 satisfy $-100 \leq D_i \leq 100$.
Test cases 9-11 satisfy $X = 10^9$.
Test cases 12-20 satisfy no additional constraints.

Languages: C++17

Source File:

Problem 4 (46480)

Problem 2-4: Bessball Hall of Fame: Report Error
Bessball is a baseball-like game that Farmer John's cows enjoy playing when they are not solving algorithmic problems.

A group of N sports-fan cows ($1 \leq N \leq 10$) all agree on who the top P Bessball players ($1 \leq P \leq 100$) are: conveniently enough, these top players happen to be the ones with uniforms numbered 1 through P.

The i-th fan has X_i opinions $(1 \leq X_i \leq 10^4)$, each of which is about a different pair of these players, and each of which takes the form "player A is better than player B".

It is guaranteed that for each fan, there is at least one ranking of all P players that is consistent with all of their opinions. It is also guaranteed that for every player, at least one fan has an opinion involving that player. (You can always count on a sports fan to have an opinion!)

Either find a single unique ranking of all P players that all the fans agree with, or determine whether there are *no* such rankings or *multiple* such rankings.

INPUT FORMAT (input arrives from the terminal / stdin):

The first line contains T, the number of subcases in the problem.

The first line of each subcase contains N and P.
The second line of each subcase contains N values X_i.

The i-th of the remaining N lines of each subcase contains $2X_i$ values $A_{i1}, B_{i1}, A_{i2}, B_{i2}, ..., A_{iX_i}, B_{iX_i}$. Interpreting these as pairs, these mean that fan i thinks player A_{i1} is better than player B_{i1}, player A_{i2} is better than player B_{i2}, and so on.

OUTPUT FORMAT (print output to the terminal / stdout):

For each subcase, print one line:
* If there is a single unique ranking that all of the fans would agree with, output these values in order from highest-ranked to lowest-ranked.
* If there is no ranking that all of the fans would agree with, output NONE.
* If there is more than one ranking that all of the fans would agree with, output MULTIPLE.

SAMPLE INPUT:

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

SAMPLE OUTPUT:

```
MULTIPLE
3 1 2
NONE
```

SCORING:

In test cases 2-5, $P \leq 8$.
In test cases 5-7, $A_{ij} < B_{ij}$, for all i and j.
In test cases 8-10, no subcase has the answer MULTIPLE.
Test cases 11-20 satisfy no additional constraints.

Languages: C++17

Source File:

Problem 5 (46612)

Problem 2-5: Maximum Exhaustion: Report Error

Farmer John's farm has N fields $(2 \leq N \leq 10^5)$ that are connected by M directed roads $(1 \leq M \leq 3 \cdot 10^5)$. The i-th of these paths goes from field A_i to field B_i and costs C_i $(-10^9 \leq C_i \leq 10^9)$ energy units to use. (Paths with negative costs actually restore energy!)

For any two fields, it is guaranteed that there is at most one directed road from the first to the second. It is also guaranteed that there are no cycles in the graph -- that is, if there is a way to get from some field x to some field y, there is not a way to get from y to x.

Bessie is at field 1 and wants to get to field N, and it is guaranteed that there is at least one way for her to do so. But Bessie is bored of always being efficient, and today she wants to find a path that spends the *most* energy.

INPUT FORMAT (input arrives from the terminal / stdin):

The first line contains N and M.
Each of the next M lines represents a directed road and contains A_i, B_i, and C_i.

OUTPUT FORMAT (print output to the terminal / stdout):

Print one line with the maximum total energy Bessie can spend.

SAMPLE INPUT:

```
4 5
2 4 9
3 2 -8
1 3 4
1 2 5
3 4 11
```

SAMPLE OUTPUT:

```
15
```

SCORING:

Test cases 2-10 satisfy $N \leq 10^4, M \leq 2 \cdot 10^4$.
Test cases 2-4 and 11-13 satisfy $C_i \geq 0$.
Test cases 14-20 satisfy no additional constraints.

Languages: C++17

Source File:

Problem 6 (46367)

Problem 2-6: Dungeon: Report Error
A castle has rooms numbered 1 through N $(1 \leq N \leq 500)$, connected by M bidirectional hallways $(1 \leq M \leq 124750)$. The i-th hallway directly connects rooms A_i and B_i $(1 \leq A_i < B_i \leq N)$, and costs W_i coins $(1 \leq W_i \leq 10^6)$ to use. Each pair of rooms is directly connected by at most one hallway.

To enter a room, an adventurer's power must be at least as large as the number of that room. To increase their power, an adventurer may spend D coins $(1 \leq D \leq 10^6)$ at any time to instantly increase their power by 1; they may do this as many times as they wish (as long as they can afford it).

Your task is to answer Q independent queries $(1 \leq Q \leq 10^5)$ of the following format: suppose there is an adventurer of level L_i $(1 \leq L_i \leq N)$, who starts in room S_i $(S_i \leq L_i)$, and the cost to gain one power is D_i. What is the fewest coins needed for this adventurer to be able to enter a different room F_i $(1 \leq F_i \leq N, S_i \neq F_i)$?

INPUT FORMAT (input arrives from the terminal / stdin):

The first line contains N, M, Q.

Each of the next M lines represents a hallway and contains A_i, B_i, W_i.

Each of the next Q lines represents a query and contains L_i, S_i, D_i, F_i.

OUTPUT FORMAT (print output to the terminal / stdout):

For each query, print one line with either -1 (if the adventurer cannot enter room F_i) or the minimum number of coins needed.

SAMPLE INPUT:

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

SAMPLE OUTPUT:

```
16
13
2
16
```

SCORING:

In test cases 2-3, $F_i = N$.
Test cases 3-6 satisfy $N \leq 50$.
Test cases 2, 4, 6, 8 satisfy $Q \leq 100$.
The other cases satisfy no additional constraints.

Languages: C++17

Source File:

Problem 7 (46495)

Problem 2-7: Bridge Town: Report Error

London is a city of N neighborhoods ($2 \leq N \leq 20000$) connected by M two-way bridges ($1 \leq M \leq 2 \cdot 10^5$). It is guaranteed that every neighborhood is reachable from every other, and that any two neighborhoods have at most one bridge between them.

Today, Phoebe has social engagements all over London, but she is worried because she has heard that the bridges there sometimes fall down.

We will call a bridge *important* if, by falling down, it would disconnect at least one neighborhood from the rest of the graph. Count the number of important bridges.

(Hint: We recommend modifying a depth-first search to keep track of some extra information that can tell you whether an edge is a bridge.)

INPUT FORMAT (input arrives from the terminal / stdin):

The first line contains N and M.

Each of the next M lines is of the form A_i, B_i, ($1 \leq A_i < B_i \leq N$) indicating that there is a bridge between neighborhoods A_i and B_i.

OUTPUT FORMAT (print output to the terminal / stdout):

For each query, print one line with the answer.

SAMPLE INPUT:

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

SAMPLE OUTPUT:

```
2
```

SCORING:

Test cases 2-5 satisfy $N \leq 2000$.
Test cases 2-9 satisfy $M \leq 40000$.
Test cases 10-20 satisfy no additional constraints.

Languages: C++17

Source File:

Problem 8 (47181)

Problem 2-8: War Shall Floyd: Report Error

Farmer John was excited to learn about the Floyd-Warshall algorithm, and he tried implementing it on a directed graph with N vertices $(4 \le N \le 500)$ and nonnegative edge costs (all between 0 and 10^9, inclusive).

Unfortunately, as many folks do, he got the loop order wrong! Specifically, the order (from outer to inner loop) is supposed to be $k, i(\text{rows}), j(\text{columns}),$, but Farmer John thought that the i, j, k should go in alphabetical order and implemented it as $i(\text{rows}), j(\text{columns}), k$. Otherwise, he implemented the algorithm correctly.

Farmer John has given you his results: an $N \times N$ grid of values A_{ij}, each of which is either an integer between 0 and 10^9 (inclusive), or 10^{18}, denoting infinity. Unfortunately, FJ can't find the edge weights for the original graph.

Your task is to find the results he should have gotten if he had implemented Floyd-Warshall correctly! (As in FJ's given grid, use 10^{18} to denote infinity.)

INPUT FORMAT (input arrives from the terminal / stdin):

The first line contains N.
Each of the next lines represents the i^{th} row of the grid and has values $A_{i1}, ..., A_{ij}$.

OUTPUT FORMAT (print output to the terminal / stdout):

Print N lines of N values each, representing the correct results.

SAMPLE INPUT:

```
4
0 1000000000000000000 2 1
1 0 3 2
1 1 0 2
2 2 1 0
```

SAMPLE OUTPUT:

```
0 3 2 1
1 0 3 2
1 1 0 2
2 2 1 0
```

SCORING:

In test cases 2-5, it is guaranteed that each edge in the original graph had a unique power-of-2 cost between 1 and 2^{30}, inclusive.
In test cases 2-9, $N \le 10$.
Test cases 10-20 satisfy no additional restrictions.

Languages: C++17

Source File: