Irish Collegiate Programming Competition 2024

Problem Set

ACM Student Chapter, University College Cork

March 2, 2024

Instructions

Rules

- All laptops, and other electronic devices must be powered off and stowed away for the duration of the contest. Mobile phones should also be stowed in the provided ziplock bags and left on your desk for the duration of the competition. Use of custom hardware (keyboard, mouse, ...) is not permitted.
- Reference books, personal notes or sheets are not permitted. You will be provided with the documentation of the supported languages.
- The only networked resource that teams are permitted to access is the submission system.
- No multi-threading is allowed, and no sub-processes.
- Use of non built-in libraries and modules, such as Python NumPY, are not supported by the server.
- No file input/output is allowed. All input to your program will be provided via standard input (stdin) and all output should be printed to standard output (stdout). Examples of how to do this in each of the languages is provided in the resources section of the submission site.
- If a team discovers an ambiguity or error in a problem statement, they should submit a question in the submission system. If the organisers agree that an ambiguity or error exists, a clarification will be issued to all teams.
- Toilet breaks: You will be allowed to leave the exam room to go to the toilet (but please go beforehand). Raise your hand and wait for an organiser to come to you.

Submission Instructions

- Your password will be provided by the organisers. Notify an organiser if you are unable to log in.
- Submissions should consist of a single source file, **not a compiled executable**.
- To submit, click the "Submit a Solution" link, complete the submission form, upload your source file, and click "Save". Your submission should now be listed in your submission queue.
- Java solutions should be a single source file and should not include the package header. The main class name should match the name of the task; for example, for the task "adding_integers", the main class should be defined as: `public class adding_integers {...}`
- C (C11) and C++ (C++17) submissions will be compiled with the flags `-lm -O2`.

Testing and Scoring

- Solutions should be submitted in the form of source code, which will be compiled on the test environment.
- The output from your program should be terminated by a new line and should not contain any additional whitespace at the beginning or end of lines.
- Except if specified otherwise in a problem's text, each solution will be tested against 10 separate test cases. Some are designed to test the limits and corner cases of the input. Test cases are not necessarily ordered in any way. Ten points are given for each correct output.
- Programs are limited to the CPU time and RAM specified within the question's text. These limits hold for each test case (compilation is not included in those restrictions).
- If a solution is submitted while an earlier solution to the same problem is in the queue, the earlier submission will not be tested and will be marked with the message: "Old, not going to be tested".
- A teams total score is based on the highest scoring submission for each question, regardless of lower scoring subsequent submissions.
- Tiebreaker. The winning team will be the one whose last scoring submission was submitted earliest.

Contributors

The UCC ACM student chapter would like to warmly thank the following people for helping with the question writing:

- Guillaume Escamocher, Insight Centre for Data Analytics
- Andrew Nash, CRT Advance
- Andrea Visentin, Insight Centre for Data Analytics
- Cadence and Keelvar Teams for providing problems and ideas

and Simone Innocente for helping with the testing phase.

1 Ciara Countless Calculation Corrections (CPU:1.0sec - RAM:16MB)

Ciara's got his dream job! Being a math teacher! He loves shaping kids minds and encourage them to pursue SEFS disciplines. However, he didn't realise the amount of effort required for correcting all the assigments produced by his students. After spending sleepless nights marking, he decided to automatize the process using computer programming.

Last week the students did a really long test on integer operations. He would like a software that automatically marks the tests, so he can happily play Zelda all weekend instead of correcting them. Your goal is to create it for him!

Each test contains N integer operations done by the students. The possible operations tested are: sum ('+'), subtraction ('-'), multiplication ('*') and integer division ('//'). Your goal is to check the percentage of correct operations done by the student and assign the correct grade (A, B, C, D, E or F).

Consider s the percentage of correct answers. The marking table is the following:

Grade	Correct answers
A	$s \geq 70\%$
B	$60\% \geq s < 70\%$
C	$50\% \geq s < 60\%$
D	$40\% \geq s < 50\%$
E	$30\% \geq s < 40\%$
F	$s < 30\%$

Input The input consists of the following lines:

- A line consisting of a single integer N, with $0 \leq N \leq 100000$, which indicates the number of operations.
- Followed by N lines, each containing one operation in the format a_i, op_i (+, -, *, or //), b_i, '=', r_i for operation i, with $0 \leq a_i, b_i, r_i \leq 100000000$. For example, line "2 + 3 = 5" has $a_i = 2$, $op_i = '+'$, $b_i = 3$, $r_i = 5$.

Output The output should consist of a single line containing the grade.

Sample Input 1

```
4
2 + 2 = 4
3 - 5 = -2
2 * 5 = 10
8 // 3 = 2
```

Sample Output 1

```
A
```

Sample Input 2

```
3
2 + 2 = 5
1 * 1 = 1
7 - 7 = 7
```

Sample Output 2

```
E
```

2 Team Travel Terrible Tickets (CPU:1.5sec - RAM:128MB)

Your organisation started using a new travel agency called "Team Travel". You suspect that the agency is not suggesting the cheapest options when booking a flight. So, you want to create your personalised flight search to prove that your company is spending a lot more for no reason.

You have a list of N quotations sent by "Team Travel" to your colleagues; all the quotations are direct flights. You downloaded a list of all M flights in the region. For each quotation, you need to search for alternative options among the available flights. You don't want to compromise your colleagues' comfort too much, so a maximum of 1 layover is allowed, e.g. you don't want them to take more than 1 connection flight. The amount of savings for each trip is the difference between the cost of the quotation provided and the cost of the cheapest option. If the quotation is the cheapest (or the only) option, the saving for that flight is 0. **N.B. In your planning, you can use "Team Travel" quotations as available flights as well.**

All airports are represented by their IATA code, a 3 capital letters code uniquely identifying an airport, e.g. Cork Airport code is ORK.

Compute how much your company can save by ditching "Team Travel".

Input The input consists of the following lines:

- A line consisting of a single integer A, with $0 \leq A \leq 10000$, which indicates the number of airports considered.
- A line consisting of a single integer N, with $0 \leq N \leq 10000$, which indicates the number of quotations provided by "Team Travel".
- A line consisting of a single integer M, with $0 \leq M \leq 100000$, which indicates the number of flights you managed to find on the web.
- Followed by N lines, each containing d_i and a_i the IATA codes of respectively the departure and arrival airport, and p_i the price for quotation i, with $0 \leq p_i \leq 10000$.
- Followed by M lines, each containing d_j and a_j the IATA codes of respectively the departure and arrival airport, and p_j the price for flight j, with $0 \leq p_j \leq 1000$.

Output The output should consist of a single line containing an integer representing the money that can be saved using cheaper options

Sample Input 1

```
5
2
5
ORK BER 200
DUB JFK 1000
ORK JFK 500
JFK DUB 100
ORK VCE 50
VCE BER 70
DUB JFK 800
```

Sample Output 1

```
280
```

The first quote is Cork (ORK) to Berlin (BER) for 200 €. The cheapest option is to do a layover in Venice (VCE), using ORK - VCE 50 € and VCE - BER 70 € for a total of 120 €. The savings on the first quotation are 200 - 120 = 80 €.

For the second quotation, there is a cheaper direct flight Dublin (DUB) to New York (JFK) for 800 €, saving 200 €.

The total savings are 280 €.

Sample Input 2

```
4
2
4
MAD ORK 150
MAD DUB 800
MAD DUB 600
DUB MAD 50
ORK MAD 70
ORK DUB 200
```

Sample Output 2

```
450
```

The first quote is Madrid (MAD) to Cork (ORK) for 150 €. There aren't other options to reach Cork from Madrid, so the savings are 0 €.

For the second flight, Madrid (MAD) to Dublin (DUB) for 800 €. There is a cheaper direct flight, for 600 €. But an even cheaper option is doing a layover in Cork; doing MAD - ORK 150 € and ORK - DUB 200 € for a total of 350 €. So the savings are 450 €.

3 Keelvar's Key Kargo

(CPU:2sec - RAM:256MB)

Kevin, a computer science PhD candidate, has decided to take up ocean freight to relax. His plan is sound: buy inventory in Kinsale, and ship it to Kyoto. Two companies offer ships on that route: one is fast but expensive, the other is slow but cheap. Both boats have a total weight capacity C, and Kevin can only afford to pay for one boat per trip.

To start off Kevin buys N items, each with a weight W_i and a value V_i. Each item is unique and cannot be packed more than once. To his dismay, the inventory worth changes by a fixed amount R_i each day. For example, fresh fruit decays quickly and could be worthless by the time it reaches its destination, while precious metals remain constant. Some goods, like whiskey, may even increase in value on the journey. Kevin is a sharp businessman and able to sell his entire cargo the day the ship arrives - and once any items reach negative value, he simply discards them.

Kevin is contractually obliged to sell his cargo in Kyoto, and so he must choose between the fast and slow ships. This means that his overall profit (i.e. the sum of all items sold minus the cost of the ship) may sometimes be negative. Before he sets sail, Kevin wants to know the maximum possible profit he can make. Can you help him?

Input The input consists of the following lines:

- A line consisting of a single integer C, with $1 \leq C \leq 1000$, which indicates the total weight capacity of the ships.
- A line consisting of two integers P_f, with $0 \leq P_f \leq 10000$, and L_f, with $0 \leq L_f \leq 365$, which indicate respectively the price of the fast ship and the number of days it takes to reach its destination.
- A line consisting of two integers P_s, with $0 \leq P_s \leq 10000$, and L_s, with $0 \leq L_s \leq 365$, which indicate respectively the price of the slow ship and the number of days it takes to reach its destination.
- A line consisting of a single integer N, with $0 \leq N \leq 1000$, which indicates the number of items available as potential cargo.
- Followed by N lines, each describing an item. Each of these N lines contains three integers
  - V_i (where $0 \leq V_i \leq 1000$), the initial value of the item.
  - R_i (where $-100 \leq R_i \leq 100$), the amount the item's worth changes by day.
  - W_i (where $1 \leq W_i \leq 10000$), the weight of the item.

Output The output should consist of a single line containing an integer equal to the maximum possible profit, $-100000 \leq A \leq 100000$.

Sample Input 1

```
50
100 3
75 7
3
100 -20 25
150 -25 20
200 -10 30
```

Sample Output 1

```
145
```

Sample Input 2

```
50
85 4
15 9
4
125 0 14
80 -4 12
50 -2 5
65 -5 18
```

Sample Output 1

```
206
```

.

4 Midleton Mud Madness

(CPU:1sec - RAM:256MB)

In October 2023, Midleton (a small town close to Cork) was completely flooded. After the water level returned to the usual level, a great amount of mud was left behind, covering streets and buildings. Everyone is trying to remove it by shovelling it out, but the amount is massive. The local authorities managed to provide a team with an excavator to remove the mud. However, the excavator's behaviour is quite strange. It can not be moved freely but has a given route that can't be changed.

You have never been good at shovelling or cleaning, but you want to help the residents restore Midleton to its beautiful condition. So, you decided to write software to help the poor Midletonians.

Midleton has N different districts with different mud quantities to remove. District i has d_i tons of mud (level of mud) to be removed.

The objective is to clean the city in a maximum of M hours. The excavator schedule is planned for all the M hours. At hour j, if the excavator is in district i, then $e_j = i$. When the excavator is in a district, the team can remove all the mud in one hour. It is also possible to clean the mud by shovelling, but that is considerably slower than using the excavator. Every hour, only one ton of mud can be removed by shovelling, but the shovelling can be done even without the excavator.

After removing all the mud, a district needs to be cleaned. That takes a full hour of the team. A district must be cleaned even if it starts with 0 mud on it.

To sum up, in every hour j with $1 \leq j \leq M$, your team can do one of the following four actions:

1. Remove all the mud in district i if the excavator is in i at time j, e.g. $e_j = i$.
2. Reduce the mud level of a district by 1, regardless of the position of the excavator. The minimum mud level is 0.
3. If the mud level of a district is 0 at that time, clean out the district.
4. Do nothing.

Your goal is to find the minimum amount of hours needed to clean all districts of Midleton, or, if it's not possible to clean it in M hours, let local authorities know.

Input The input consists of the following lines:

- A line consisting of a single integer N, with $1 \leq N \leq 10000$, which indicates the number of districts in Midleton.
- A line consisting of a single integer M, with $1 \leq M \leq 1000000$, which indicates the maximum number of hours used for the cleaning.
- Followed by N lines, each containing d_i the level of mud (tons of mud) to be removed from district i, with $0 \leq d_i \leq 10000$.
- Followed by M lines, each containing e_j the district in which the excavator is at hour j with $1 \leq e_j \leq M$.

Output The output should consist of a single line containing an integer representing the minimum amount of hours needed to clean the city. If it's not possible to clean Midleton in M hours, return -1.

Sample Input 1

```
3
7
6
2
0
3
3
1
3
3
3
3
```

Sample Output 1

```
6
```

The mud levels are $d = [6, 2, 0]$. An optimal solution is the following:

1. Clean district 3.
2. Shovel the mud in district 2, reducing it from 2 to 1.
3. Use the excavator to remove all mud in district 1, since $e_3 = 1$.
4. Clean district 1.
5. Shovel the mud in district 2, reducing it from 1 to 0.
6. Clean district 2.

So, 6 hours in total.

Sample Input 2

```
2
5
6
6
1
1
1
1
2
```

Sample Output 2

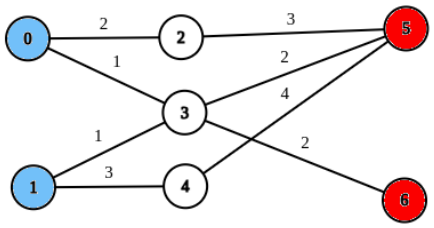
```
-1
```

The mud levels are $d = [6, 6]$. Removing the mud and cleaning all the districts in 5 hours is impossible.

5 Cadence's Careful Circuit Construction (CPU:1.5sec - RAM:512MB)

Cadence is releasing a new tool to optimise its chip designs, the *Cadence Careful Circuit Construction* (CCCC). Its goal is to reduce the processor's cost of production to increase their customers' margin. In a chip, all the source pins must be connected with non-overlapping paths to the destination pins. These paths are built with a special new superconductive alloy, the *Costinum*. This is the most expensive material in chip production. The goal of your tool is to compute the cheapest way to connect all source pins with the destination ones using the minimum amount of *Costinum*.

The chip substrate can be seen as an undirected weighted graph. In this graph, we have k nodes that represent the input pins, and another (non-intersecting) set of k nodes are designated as output pins.



You need to create the optimiser for the CCCC. Its goal is to find a minimal set of k *routings* - paths from inputs to outputs. Each input can be matched to any output, so long as each input and output are only used on one routing. None of the routings can share an edge. Each edge of the graph i has an associated cost of filling it with *Costinum*, w_i. Two paths can't use the same node since they can't touch. A technological innovation allows us to create an "overpass" where multiple paths are using the same node. However, this requires an additional P penalty cost in *Costinum* that must be paid for each path that uses that shared node. For the R^{th} routing through a node, a penalty of $P \times (R-1)$ is incurred. For example, given a node, on the first routing through this node a penalty of $P \times (1-1) = 0$ is incurred. If two additional routings overpass the node, they will incur a penalty of $P \times (2-1) + P \times (3-1) = P + 2P = 3P$. Note that it **is** possible to overpass an input or output node in exactly the same manner as any other node.

The cost of a routing is the sum of the weights on its edges plus any penalties incurred. The total cost of the chip is the sum of all the k routings.

Constraints

- $2 \leq N < 2\,500$, the number of nodes on the chip
- $1 \leq E < N^2$, the number of edges on the chip
- $1 \leq K < 50$, the number of inputs and outputs to the circuit
- $0 \leq P < 500$, the penalty for each additional (after the first) routing using a given node
- $1 \leq W_i < 500$, the weight on edge i

Subtasks

- **Subtask 1 (5 points):** $K = 1$, all $W_i = 1 \quad \forall \quad i$ I.e., there is only one input and output to the circuit, and all weights are 1
- **Subtask 2 (15 points):** $K = 1$ I.e., there is only one input and output to the circuit, but the weights can be of any value
- **Subtask 3 (10 points) :** $N < 500, P = 0$
- **Subtask 4 (20 points):** $N < 500$, $P > 100000000$, it is guaranteed that there is an on optimal set of k routings that don't overpass a node
- **Subtask 5 (50 points):** $N < 500$

Input

- The first line contains a pair of space separated integers, N and E
- The second line contains a single integer, P
- The following line contains k space separated integers - the designated input nodes
- The following line contains k space separated integers - the designated output nodes
- E lines follow, each containing space-separated integers f_i, t_i, W_i - indicating that line i corresponds to an edge of weight W_i between nodes f_i and t_i

Output

Output a single integer, the minimum cost of the k routings on the chip

Sample Input 1

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

Sample Input 2

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

Sample Output 1

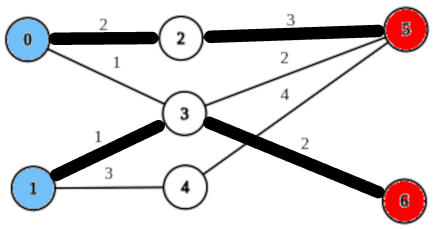
```
8
```

Sample Output 2

```
7
```

Explanation of Sample input/output

Sample Input 1 The following is the optimal set of two routings from the nodes $1, 0$ to $5, 6$



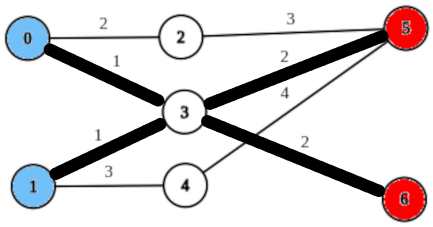
Sample Input 2 Sample input 2 contains the same graph - but since the penalty is 1, it becomes more efficient to take a penalty by routing through node 3 twice

Explanation of subtasks

For some tasks, we have divided the test cases into **subtasks**.

Each subtask is allocated a certain number of points - for this particular problem, the subtasks are worth 5,15,10, 20 and 50 points respectively.

To get the points for a sub-task, your solution must solve **all** of the test cases in that particular sub-task within 1 second. E.g., solving 2/5 cases for subtask 2 scores 0 points, where solving 5/5 cases will score 15 points.



An important detail is that you do not have to (but can if you wish) solve all of the sub-tasks in the same submission - e.g. if you make a submission that solves Subtask 1 and scores 5 points, and a separate solution that solves only subtask 2 and scores 15 points - **your total score on that problem will be 20 points**, and so on.