Irish Collegiate Programming Competition 2025

Problem Set

ACM Student Chapter, University College Cork

March 1, 2025

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:

- Cadence, Keelvar and F5 NGINX Teams for providing problems and ideas
- Guillaume Escamocher, Insight Centre for Data Analytics
- Andrew Nash, ADVANCE CRT
- Andrea Visentin, Insight Centre for Data Analytics
- Dr Sabin Tabirca, Department of CS

as well as , , and for helping with the testing phase.

1 Ton of Tariffs

(CPU:0.5sec - RAM:64MB)

In an increasingly protectionist world, countries impose tariffs on imported goods to protect domestic industries and generate revenue. However, these tariffs have a direct impact on trade volume. If tariffs are too high, trade dwindles; if too low, domestic industries suffer from foreign competition. You have been hired as an economic analyst for the Global Trade Federation to calculate the impact of tariffs on trade and the revenue generated.

International trade operates on a simple principle: countries exchange goods based on supply and demand, but tariffs influence these exchanges. Historically, nations have implemented varying tariff policies, often leading to trade wars or economic booms. Your task is to compute the expected tariff revenue based on current trade regulations. However, tariffs not only increase costs but also reduce the total weight of goods traded. The weight of goods traded decreases linearly with the imposed tariff. If the tariff reaches 100%, only half of the original trade volume occurs, and at 200%, trade ceases altogether. Different tariffs can be implemented for different product categories. The effective weight of products traded for each category, W_{eff}, is given by:

$$W_i^{eff} = W_i \times \max\left(0, 1 - \frac{T_i}{200}\right) \tag{1}$$

The tariff on a product does not influence the weight of the other ones. The total tariff revenue for product i, R_i, is calculated as:

$$R_i = W_i^{eff} \times P_i \times \frac{T_i}{100} \tag{2}$$

where for each product i:

- W_i is the original weight in tonnes.
- T_i is the tariff rate in percentage.
- P_i is the price of the product per tonne.
- W_i^{eff} is the adjusted weight after considering tariff effects.

Input The input consists of the following lines:

- A line consisting of a single integer N, with $1 \leq C \leq 1000$, which indicates the total amount of products.
- Followed by N lines, each describing a product category (i). Each of these N lines contains three integers
  - W_i (where $0 \leq W_i \leq 10000$), the weight in tonnes.
  - T_i (where $0 \leq T_i \leq 200$), is the tariff rate in percentage.
  - P_i (where $1 \leq P_i \leq 10000$), is the price of the product per tonne.

Output Print a single floating-point number, the total tariff revenue, rounded to two decimal places.

Sample Input 1

```
2
100 10 1000
500 200 1600
```

Sample Output 1

```
9500.0
```

Sample Input 2

```
4
3227 21 5730
6214 149 7661
2192 202 3126
4930 45 995
```

Sample Output 2

```
23273762.00
```

2 Simple Successor Search (CPU:0.5sec - RAM:512MB)

The goal of this task is to return the successor of a given number. The successor of a number is the number plus one. So the successor of zero is one, the successor of one is two, the successor of two is three, and so on.

All numbers given are whole numbers. The smallest number that can be part of the input is zero, and the largest is nine hundred and ninety-nine. Consequently, all numbers in the output should be at least one and at most one thousand.

Input The input consists of two lines, each consisting of one number. All numbers are composed only of lowercase letters, so no space, hyphen, or other special character.

Output The output shall consist of two lines:

- a line consisting of the successor of the number given in the first line of the input
- a line consisting of the successor of the number given in the second line of the input

Numbers in the output should be written in the same format as in the input, with only lowercase letters.

Sample Input 1

```
threehundredandfortynine
ninetynine
```

Sample Output 1

```
threehundredandfifty
onehundred
```

Sample Input 2

```
zero
ninehundredandninetynine
```

Sample Output 2

```
one
onethousand
```

3 Cadence Chip Composer (CPU:0.5sec - RAM:64MB)

The world of semiconductor manufacturing is highly competitive, where efficiency and precision dictate success. Cadence Design Systems, a market leader in hardware, software, and IP for electronic design, wants to build a new software tool called **Chip Composer**, for optimising the layout of electronic designs on chip. The software must arrange rectangular components optimally on a microchip, ensuring maximum utilization of the available space while minimizing wastage and improving energy efficiency.

Modern microchips are intricate assemblies of millions of transistors and electronic components, requiring careful placement to optimize performance and reduce power consumption. The challenge lies in arranging these components to fit within the chip's fixed dimensions while maintaining a high packing efficiency. The goal is to pack all rectangular components onto a rectangular chip without overlap, while also measuring the efficiency of the layout. Each component has a fixed width and height, and you must ensure that:

- All components fit within the chip's dimensions.
- No components overlap.
- Compute the efficiency. The efficiency represents the percentage of the chip that is occupied:

$$\text{Efficiency} = \frac{\text{Total Area of Placed Components}}{\text{Total Chip Area}} \times 100$$

Input The input consists of the following lines:

- A line consisting of two integers W and H (where $1 \leq W, H \leq 1000000$), the width and height of the chip.
- A line consisting of a single integer N, with $1 \leq N \leq 1000000$, representing the number of components in the chip.
- Followed by N lines, each describing a chip component (i). Each of these N lines contains four integers (x_i, y_i, w_i, h_i):
  - x_i, y_i (where $1 \leq x_i \leq W - w_i$, and $1 \leq y_i \leq H - h_i$), are the coordinates of the top-left corner.
  - w_i, h_i (where $1 \leq w_i \leq W$, and $1 \leq h_i \leq H$), are the width and the height of the component.

Output If all components fit without overlap, print the efficiency percentage rounded to two decimal places. Otherwise, print -1.

Sample Input 1

```
10 10
3
1 1 4 4
5 1 3 3
7 7 2 2
```

Sample Output 1

```
29.00
```

Sample Input 2

```
10 10
3
1 1 4 4
4 1 4 4
4 4 2 2
```

Sample Output 1

```
-1
```

4 Batches of Big Boxes

(CPU:.5sec - RAM:512MB)

In the country of Cephal, all cities are planned out in the same way. In every Cephalian city, all streets start from the same central location and extend outwards. An example of a Cephalian city with three streets and sixteen houses is shown in Figure 1. In this city, the first street extends to the northeast, the second street to the southeast, and the last street to the west.

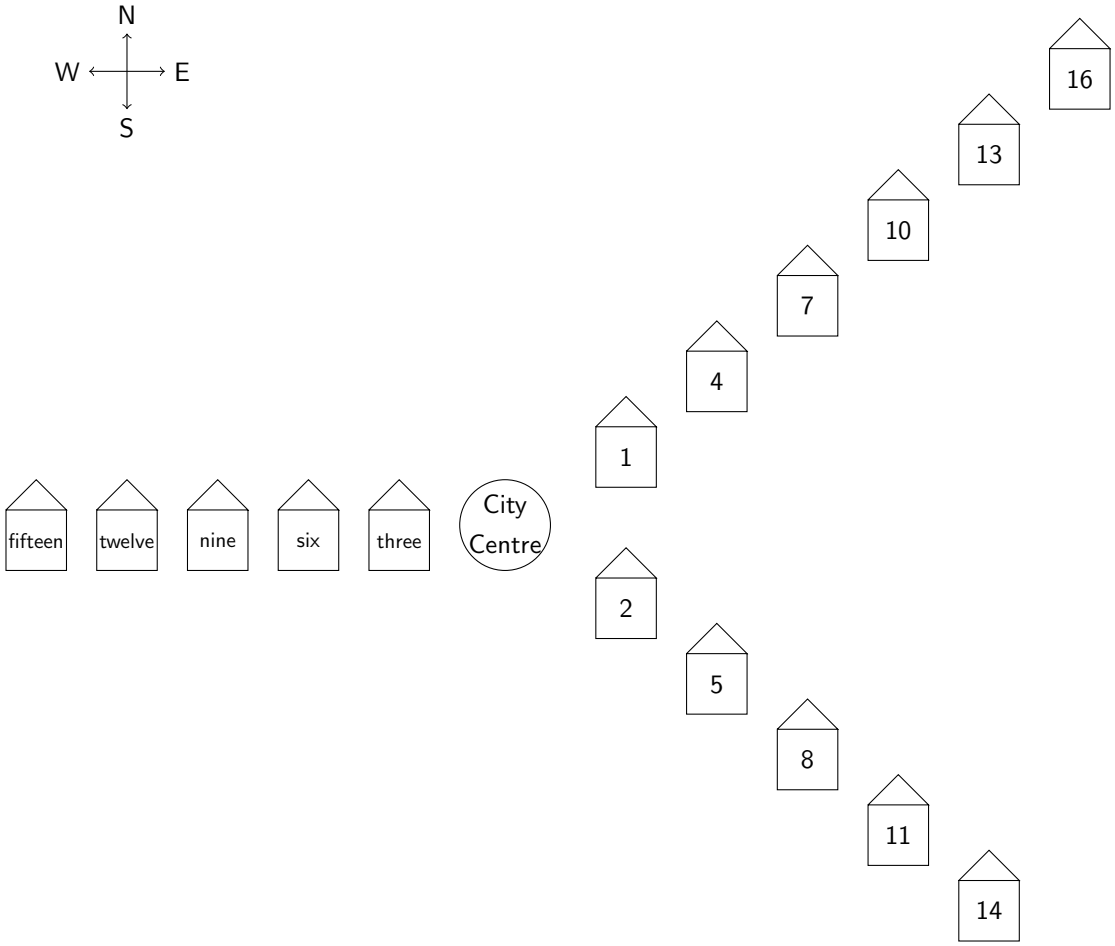

Figure 1: Plan of a Cephalian city.

House numbering in Cephal is done city-wide instead of street-wide. This means that if a city contains n houses, then every number from 1 to n will be assigned to exactly one house in this city. The distribution of these house numbers can be seen as roughly following the shape of a spiral that originates from the city centre. If a city has k streets, then numbers 1 to k will be assigned to the house from each street that is closest to the city centre, numbers $k+1$ to $2k$ will be assigned to the house from each street that is second closest to the city centre, and so on. The i^{th} street of this city will contain houses with numbers i, $k+i$, $2k+i$,.... The further apart a house is from the city centre, the higher its number will be. For example, if a city has 6 streets and 45 houses, then the houses in the third street will be assigned, in increasing order of distance from the city centre, numbers 3, 9, 15, 21, 27, 33, 39, and 45, and the houses in the fourth street will be assigned numbers 4, 10, 16, 22, 28, 34, and 40.

Most house numbers are represented with digits on the houses' doors. However, houses in the last street of each city have traditionally fully spelled out their numbers on their doors. For example, in a city with k streets, a house with number 12 would put the digits 1 and 2 on its front door if 12 is not a multiple of k, but would instead put the letters 't', 'w', 'e', 'l', 'v', and 'e' again if 12 is a multiple of k. See Figure 1 for an illustration of the latter. A recent population increase in the country has made this peculiar custom difficult to apply, because in some large cities, the representation of house "numbers" does not fit on the doors. For this reason, the president of Cephal has issued an executive order to replace letters by digits in the last street of every city.

In every city, ten workshops are set up to produce the digits that will be needed. There is one workshop for every digit, so one of the workshops is tasked with producing the 0s, another is tasked with producing the 1s, and likewise for the remaining digits. Each workshop will be mailing a box to, and only to, each house in the last street of the city which number contains at least one occurrence of the digit produced by the workshop. Each box will contain many duplicates of this digit, to account for people that want to display their house numbers on multiple doors. All workshops are capable of extremely high production, and do not mind manufacturing excess digits, so they are not interested in knowing the exact number of digits that is strictly needed. All the workshops care about is the exact number of boxes to prepare. Your goal is to determine this number.

As an example, consider a city with 50 houses spread over 4 streets. The last street therefore contains houses with numbers 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, and 48. The workshop in charge of producing the digit 4 will thus need exactly 5 boxes, for houses 4, 24, 40, 44, and 48. At most one box per digit is sent to each house, even if the house number contains more than one occurrence of a digit, which is why the house with number 44 will only receive one box from the workshop that produces the digit 4. Furthermore, no number can start with a 0, so the workshop producing the digit 0 in this city will only need 2 boxes: one for house with number 20 and the other for house with number 40.

Input The input consists of ten lines $I_1, I_2, \ldots, I_{10}$, each consisting of three integers k, n and d. The first integer, k, is the number of streets in the city and ranges from 1 to 10 inclusive. The second integer, n, is the number of houses in the city and ranges from k to 10^{18} inclusive. The third and last integer, d, is a digit and ranges from 0 to 9 inclusive.

Output The output shall consist of ten lines $O_1, O_2, \ldots, O_{10}$, with Line O_i consisting of a single integer equal to the exact number of boxes that the workshop producing the digit d will need to prepare in a city with k streets and n houses, with the values for k, n and d taken from input line I_i.

Sample Input 1

```
4 50 0
4 50 1
4 50 2
4 50 3
4 50 4
4 50 5
4 50 6
4 50 7
4 50 8
4 50 9
```

Sample Output 1

```
2
2
5
2
5
0
2
0
3
0
```

Sample Input 2

```
8 22 2
1 338 2
1 128 4
1 25 2
2 100 9
3 454 4
4 360 4
5 242 6
6 718948 1
1 418136 9
```

Sample Output 2

```
0
151
22
8
5
43
26
4
53006
169433
```

5 Keelvar Klever Kosmic Kargo

(CPU:1sec - RAM:256MB)

The year is 2200, and interstellar commerce is at an all-time high. Keelvar is one of the premier companies specializing in the scheduling of expensive deliveries across the vast expanse of space. However, space travel is expensive! Fuel costs and cosmic warming are major concerns. Keelvar needs a route optimization algorithm that finds the cheapest way to deliver a package across multiple planetary systems.

Space travel follows predefined hyperspace routes connecting different stars. Each route between two stars has an associated cost. Ships can travel to the next star in sequence, or they can use a powerful booster to skip an intermediate star, albeit at a higher cost. To complicate matters further, unstable blackholes exist along the way. These blackholes act as wormholes, allowing for rapid travel to further locations, but it is uncertain which one a ship will emerge from.

The journey consists of a sequence of stars aligned in a straight path, from start (star 1) to destination (star N). Each star provides three travel options:

- **Standard Jump**: Move to the next star (cost provided).
- **Booster Jump**: Skip the next star and move two steps forward (higher cost but faster).
- **Blackholes**: Some stars contain a single blackhole that teleports you forward but has two random exits with different probabilities and associated costs.

Your task is to compute the minimum expected cost to transport a package from the starting point to its destination, considering all available travel options, including boosters and blackholes.

Input The input consists of the following lines:

- A line consisting of two integers N and B (where $1 \leq N, B \leq 1000000$), the number of stars in the sequence and blackholes, respectively.
- A line consisting of $N-1$ integers $c_1, \ldots, c_{N-1}$, with $1 \leq c_i \leq 10000$, where c_i is the cost to jump from star i to star $i+1$.
- A line consisting of $N-2$ integers $b_1, \ldots, b_{N-2}$, with $1 \leq b_i \leq 10000$, where b_i is the cost to use the booster and jump from star i to star $i+2$.
- Followed by B lines, each describing a blackhole (j). Each of these B lines contains six integers $(s_j, e_{j1}, p_{j1}, e_{j2}, p_{j2}, c_j)$:
  - s_j (where $1 \leq s_j \leq N-2$), is the starting star
  - e_{j1}, e_{j2} (where $s_j < e_{j1}, e_{j2} \leq N$), are the two possible exit stars.
  - p_{j1}, p_{j2} (where $0 \leq p_{j1}, p_{j2} \leq 100$ and $p_{j1} + p_{j2} = 100$), are the probabilities of landing in e_{j1}, e_{j2} respectively,
  - c_j ($1 \leq c_j \leq 100000$), is the cost to enter the blackhole.

Output Print a single number: the minimum expected cost to reach star N, rounded to two decimal places.

Special case While these conditions are valid for all the known universe. Some of the testcases are relative to 2 particular types of galaxies:

- Some galaxies are blackhole free.
- Some galaxies have deterministic blackholes. Where in each blackhole (j), one of the two exits is never used, e.g., $p_{j1} = 0$ or $p_{j2} = 0$.

Sample Input 1

```
5 1
3 2 4 6
3 5 5
1 4 5 80 20 1
```

Sample Output 1

```
5.80
```

Sample Input 2 (No Blackholes)

```
5 0
4 3 4 3
9 8 9
```

Sample Output 1

```
14.00
```

6 F5 NGINX's Fantastic Forensics (CPU:1sec - RAM:256MiB)

Web servers generate a massive amount of HTTP request logs, which are critical for monitoring and detecting suspicious activity - an integral part of critical services delivered by F5 NGINX.

Your task is to analyse some given logs containing redacted information about incoming requests to a server, including only the timestamp, source IP address, and the type of request (specifically GET, POST, PUT, DELETE).

Recently, there has been an increase in suspicious activity on this server, and your team needs to analyze the logs to identify potential threats. Specifically, you need to determine which IP addresses made some specific types of requests (GET, POST, PUT, or DELETE) within a given time range. This information will help the team investigate potential security breaches or unauthorized access attempts.

You must find and print (in lexicographic order), for different queries, all of the distinct IP addresses that made a specific type of request (GET, POST, PUT, DELETE) between (inclusive) a given pair of timestamps (of the form *YYYY*-*MM*-*DD*T*HH*:*MM*:*SS*)

Input

- A line containing integer N(where $1 \leq N \leq 50,000$), the number of lines in the log
- N lines follow, each containing a line of the logfile - a timestamp, IP address and request type, all separated by spaces. Note that the logs aren't necessarily in order of time.
- This is followed by a line containing integer Q(where $1 \leq Q \leq 500$), the number of requests.
- Q lines follow, each containing a pair of timestamps and a request type.

Output Output consists of Q lines

For each query output the IPs that satisfy the query in lexicographical order, separated by spaces, on a single line.

If there are no satisfying IPs, print the string NONE for that query

Sample Input 1

```
10
2023-10-01T12:34:56 192.168.1.1 GET
2023-10-01T12:41:10 192.168.1.8 PUT
2023-10-01T12:39:50 192.168.1.6 POST
2023-10-01T12:35:10 192.168.1.2 POST
2023-10-01T12:37:30 192.168.1.4 PUT
2023-10-01T12:36:20 192.168.1.3 DELETE
2023-10-01T12:40:00 192.168.1.7 DELETE
2023-10-01T12:38:40 192.168.1.5 GET
2023-10-01T12:42:20 192.168.1.9 GET
2023-10-01T12:43:30 192.168.1.10 POST
3
2023-10-01T12:34:56 2023-11-02T12:40:00 GET
2023-10-02T12:35:10 2023-10-02T12:42:20 POST
2023-10-01T12:36:20 2023-10-01T12:43:30 DELETE
```

Sample Output 1

```
192.168.1.1 192.168.1.5 192.168.1.9
NONE
192.168.1.3 192.168.1.7
```

7 Nuanced Nice Number Nitpicking (CPU:1sec - RAM:256MB)

Nice numbers are numbers which are formed from the digits $2, 3, 7, 9$ only and are divisible by 3.

Given an integer N, determine how many *nice* numbers there are that are exactly N digits long.

Input The input consists of a line consisting of a single integer N, with $1 \leq N \leq 20$

Output The output should consist of a single integer, the output modulo $1,000,000,007$

Sample Input 1

1

Sample Output 1

2

Sample Input 2

2

Sample Output 1

6

Explanation of sample input For sample input 1, the nice numbers are 3 and 9. For sample input 2, the nice numbers are 27,33,39,72,93 and 99.

8 NetApp Nexus Neural Navigation (CPU:2sec - RAM:256MB)

NetApp has deployed an advanced Neural Nexus system, a fleet of autonomous repair drones designed to repair critical server components across their data centers. Each server node requires periodic maintenance checks at precise 3D coordinates inside NetApp's high-tech facilities.

To perform a repair, a drone must fly directly to a precise set of 3-D coordinates (in meters) (x,y,z) and remain in position for at least 1 second while it fixes the issue. Each site has restricted time windows when repairs can be conducted due to system load balancing and access restrictions. The drone may travel to a repair site outside these periods but can only conduct repairs within the periods.

You are given N such coordinates to repair, each of which must be fully fixed during a permissible slot (i.e, the repairment must be complete **within** the slot).

You have D drones available.

- Each drone starts at a specific position inside the facility (possibly not unique) at time $t = 0$, where time is measured in units of seconds.
- Drones have a fixed speed (m/s) and a finite battery capacity (seconds).
- A drone consumes battery continuously from launch until it completes its final inspection. It can hover indefinitely, but doing so still drains battery power.
- Drones can only be launched at exactly $t = 0$. If a drone is not needed for doing any of the N repairs, it can be left off, and it does **not** consume any battery.
- After its **last** repair, a drone can be turned off on the spot and does not use any more battery. A drone turned off can **not** be turned on again. A drone does **not** have to return to the starting position. It can be left in the location of the last repair, as soon as the repair is complete.

The drone can travel directly between two points, with distance described by the euclidean norm - $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$ (we assume it can magically fly through the servers!), or hover in place for any duration of time.

You are responsible for coordinating the fleet of Nexus drones to minimize total battery consumption while ensuring all inspections are successfully completed **within** the allowed timeframes.

Input The input consists of the following lines:

- A line containing space-separated integers N and D (where $1 \leq N \leq 10$, $1 \leq N \leq 5$) - the number of repair locations and drones respectively
- Followed by N lines, each describing a repair task (i). Each of these N lines contains:
  - x_i, y_i, z_i are the repair locations. All coordinates are $0 \leq x_i, y_i, z_i \leq 100$.
  - T_i, (where $1 \leq T_i \leq 2$) an integer describing the number of permissible windows during which the location can be surveyed.
  - T_i space-separated pairs of space-separated integers t_j^s t_j^e, signifying that (t_j^s, t_j^e) is the j-th possible (where $1 \leq j \leq T_i$) period during which this location can be repaired. This location cannot be repaired outside these times.
- Followed by D lines, each describing a drone (k). Each of these D lines contains:
  - x_k, y_k, z_k are the starting coordinates of the drone.
  - s_k representing the drone speed in m/s (where $1 \leq s_k \leq 20$).
  - c_k representing the drone battery charge in seconds (where $1 \leq c_k \leq 100$).

Output Print a single float, rounded to 4 decimal places - the minimum total battery capacity consumed across all drones in the most efficient route that performs all inspections.

It is guaranteed that there is a possible routing for drones to perform all inspections.

Sample Input 1

```
2 1
1.0 2.0 3.0 1 0 5
4.0 2.0 1.0 1 7 10
0.0 0.0 0.0 1.0 15
```

Sample Output 1

```
9.3472
```

Sample Input 2

```
3 2
1.0 2.0 3.0 2 0 5 10 15
4.0 5.0 6.0 2 1 6 12 18
7.0 8.0 9.0 2 2 7 14 20
0.0 0.0 0.0 1.0 20.0
1.0 1.0 1.0 1.5 15.0
```

Sample Output 1

```
19.1962
```