# Lab: Problem Solving

This document defines the **in-class exercise** assignments for the ["Algorithms" course @ Software University](https://softuni.bg/opencourses/algorithms).

# Shelter

You are Commander Shepard and you have **S** soldiers who are located in a warzone. When the enemy fires a missile, your soldiers must hide in one of the **M** nearby shelters. However, shelters have a limited capacity **C** (they cannot hold more than **C** soldiers).

Soldiers and shelters will be given as **2D points**. Assume all soldiers can move **1 unit** of distance per second.

Write a program that finds the **shortest time** it will take for all your soldiers to hide in a shelter.

## Input

* On the first input line you will be given **S** (number of soldiers), **M** (number of shelters) and   
  **C** (shelter capacity), separated by a single space**.**
* On the next **S** lines you will be given the coordinates of each soldier in the format "**X Y**".
* On the next **M** lines you will be given the coordinates of each shelter in the format "**X Y**".

## Output

* On the only ouput line print the **shortest time** it takes for all soldiers to take shelter (in seconds, rounded to the 6th digit after the decimal point).

## Constraints

* The numbers **S** (number of soldiers), **M** (number of shelters) and **C** (capacity) will be in the range **[1..500]**.
  + In 50% of the tests **S, M, K <= 15**.
  + There will always be enough shelters for all soldiers to hide.
* All given coordinates will be integers in the range **[-1000…1000]**.
  + There may be multiple soldiers/shelters with the same coordinates.
* Time limit: **1000 ms**. Allowed memory: **50 MB**.

## Examples

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Output** | **Visual** | **Comments** |
| 5 3 2  3 14  10 8  5 9  14 8  12 4  3 12  11 7  5 13 | 7.071068 |  | One of the optimal solutions.  Soldiers with coordinates (3,14) and (5,9) go to shelter (3,12). (12,4) and (14,8) go to shelter (11,7) and soldier (10,8) goes to shelter (5,13).  Notice that (10,8) can get to shelter (11,7) more quickly, but the capacity does not allow it. |

# Fast & Furious

The ministry of interior recently deployed a system of **traffic cameras** on different locations on the roads. Some **pairs of cameras** relate to direct road and you are given the **distance** and the **speed limit** between them.

You are given the **records from the traffic cameras** on the road. Each camera takes photos of car license plate numbers and **reports the numbers and the time of observation**. Your task is to find which cars are **speeding**.

A car travelling between two arbitrary cameras **A** and **B** on the road is **speeding** if it takes the distance between these cameras for **less time than the minimum possible** within the allowed speed limits. Note that many routes may exist between **A** and **B** and each of them can be passed for different times depending on the distances and speed limits for the roads in each route. We assume that the drivers always take the fastest route between two cameras.

## Input

* The input is read from the console.
* At the first line, the word “**Roads:**” stays.
* The next few lines hold **pairs of camera names**, the **distance** between them and the **speed limit** (in km/h) between them. The camera names and maximum speed are separated by a single space. Example:

**CameraSofia CameraPleven 133.35 140**

* At the next line, the word “**Records:**” stays.
* The next few lines hold a sequence of **camera records**. Each record consists of a **camera name**, a **license plate number** and a **time** in 24-hour format (**hh:mm:ss**), separated by a space. Example:

**CameraSofia CA1111AA 12:56:12**

* The last line holds the word “**End**” only.

## Output

* Print the **license plate numbers of all speeding cars** in alphabetical order, each on separate line. Example:

**CA1111AA  
CA1212BB  
CHY0L0428**

## Constraints

* All **camera names** consist of letters and digits.
* All **license plate numbers** consist of letters and digits.
* The **distances** and **speed limits** are real numbers in the range [1…10 000].
  + The symbol “.” is used as decimal separator.
* The **number of roads** is in the interval [1; 1 000].
* The number of **camera records** is in the interval [1; 10 000].
* All data is collected on the **same day**.
* Cameras collect their records in unspecified order.
* Time limit: **200 ms**. Allowed memory: **24 MB**.

## Examples

|  |  |
| --- | --- |
| **Input** | **Visualization and Comments** |
| Roads:  Sofia Plovdiv 145.4 90  Plovdiv Varna 361.4 120.5  Varna Burgas 114.95 30  Burgas Plovdiv 252.9 42  Records:  Varna CA1234AA 19:48:25  Burgas B4732AH 19:38:50  Sofia CA1234AA 08:32:18  Plovdiv A777777 15:28:56  Varna SP33D 02:24:18  Burgas A777777 18:42:15  Plovdiv CA1234AA 15:32:18  Sofia SP33D 04:32:51  Varna B4732AH 08:18:36  End | The cameras are connected by roads like shown below:    Car “A777777” is speeding between Plovdiv and Burgas.   * It takes the distance of 252.9 km from Plovdiv to Burgas for 3:13:19 hours (18:42:15 @ Burgas - 15:28:56 @ Plovdiv) ≈ 3.222 h. * The minimum time within the allowed maximum speed limits from Plovdiv to Burgas is 252.9 km / 42 km/h ≈ 6.02 hours. * The car was speeding because the driving time (3.222 hours) **<** the minimum possible time within the speed limits (6.02 hours).   Car “SP33D” is speeding between Varna and Sofia.   * It takes the distance between Varna and Sofia for 2:08:33 hours (04:32:51 @ Sofia - 02:24:18 @ Varna) ≈ 2.1425 hours. * Two routes exist from Varna to Sofia:   + For the route Varna  Plovdiv  Sofia the minimum time within the speed limit is 2.999 hours (Varna  Plovdiv) + 1.616 (Plovdiv  Sofia) ≈ 4.615 hours.   + For the route Varna  Burgas  Plovdiv  Sofia the minimum time within the speed limit is ≈ 11.469 hours. * The car was speeding because the driving time (2.1425 hours) **<** the minimum possible time within the speed limits (4.615 hours). |
| **Output** |
| A777777  SP33D |

|  |  |
| --- | --- |
| **Input** | **Visualization and Comments** |
| Roads:  Matzoro Isterni 128.55 50  Matzoro Kostos 87.25 48.5  Isterni Kostos 100 40.52  Melanes Galanado 230.5 50  Records:  Isterni AOM5973 13:20:11  Matzoro IBK5674 08:35:12  Matzoro AHI1278 08:35:12  Galanado IBK5674 18:20:35  Kostos COM1515 05:38:02  Galanado COM1515 08:40:15  Isterni IBK5674 14:28:30  Melanes COM1515 22:31:50  Kostos AOM5973 12:46:21  Kostos COM1515 18:56:10  End | The cameras are connected by roads like shown below:    Car “AOM5973” is speeding between Kostos and Isterni:   * It takes the distance of 100 km from Kostos to Isterni for 0:33:50 hours (13:20:11 @ Isterni - 12:46:21 @ Kostos) ≈ 0.564 h. * The minimum time within the allowed maximum speed limits from Kostos to Isterni is 100 km / 40.52 km/h ≈ 2.468 hours. * The car was speeding because the driving time (0.564 hours) **<** the minimum possible time within the speed limits (2.468 hours). |
| **Output** |
| AOM5973 |

# Renewal

In Bulgaria there are road problems – too many holes! This is why in the parliament they decided it was about time to renew most of the roads (and get some money below the table, of course).

In Bulgaria there are **N** cities. Between some of the cities there are bidirectional roads. We can say that a route between two cities exists when we can start from the first city and end in the second city, no matter the cities in between we must go through.

Current situation in the country is not very good. Some cities do not have roads between them, others have more than one, which makes the traffic control difficult. Our parliament wants to build new roads and destroy others in such way that after the whole reconstruction there will be **exactly one route between each pair of distinct cities.** Since building and destroying new roads is expensive, the parliament wants to minimize the expenses for the new road network as much as possible.

You will be given a matrix of numbers **M** with size **NxN** representing the current state of the roads. There is a bidirectional road between city **i** and city **j** only if **M[i][j]** equals **1** and **M[j][i]** equals **1**. You will also be given matrices **B** and **D** also with size **NxN**. If between cities **i** and **j** there is no road, then the cost for its building will be **B[i][j]** and if there is a road, then the cost for its destroying will be **D[i][j]**. Prices in **B** and **D** are written as symbols where capital English letters **A, B, …, Z** represent the values **0, 1, …, 25** accordingly and the lower case letters **a, b, …, z** represent the values **26, 27, …, 51**. Your task is to write down a program, which by given road state calculates the minimum available sum needed for the reconstruction.

## Input

The input data should be read from the console. The input will contains exactly **3 \* N + 1** lines.

On first input line there will be the number **N**.

On the next N lines, there will be the matrix **M**.

On the next N lines, there will be the matrix **B**.

On the next N lines, there will be the matrix **D**.

The input data will always be valid and in the format described. There is no need to check it explicitly.

## Output

The output data should be printed on the console.

On the first and only output line, print the minimum sum needed for the reconstruction.

## Constraints

* **N** will be between **1** and **50**, inclusive.
* **M**, **B** and **D** will be **NxN** matrices.
* **M** will contain only **0** and **1**.
* **B** and **D** will contain only lower and upper case English letters.
* For each **i** and **j**, **M[i][j]**, **B[i][j]**, **D[i][j]** will equal **M[j][i]**, **B[j][i]**, **D[j][i]** accordingly.
* There will be no road, which starts and ends in the same city.
* Values in **B**, which correspond to existing road, can be ignored.
* Values in **D**, which correspond to non-existing roads, can be ignored.
* Allowed working time for your program: **0.1** seconds. Allowed memory: **16 MB**.

## Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Explanations** |
| 3  000  000  000  ABD  BAC  DCA  ABD  BAC  DCA | 3 | We have three cities, without any roads. We build the roads 0-1 and 1-2 – final result 3. |
| 3  011  101  110  ABD  BAC  DCA  ABD  BAC  DCA | 1 | All cities are connected to each other. We need to destroy one of the roads. The cheapest solution is 0-1. |
| 6  011000  101000  110000  000011  000101  000110  ABDFFF  BACFFF  DCAFFF  FFFABD  FFFBAC  FFFDCA  ABDFFF  BACFFF  DCAFFF  FFFABD  FFFBAC  FFFDCA | 7 | - |