Problem Solving Strategies – set of more advanced problems

Maria Keet
University of Cape Town, South Africa

ICPC Training Session, April 12, 2015

Task: categorise the following set of problems into the right strategies.

Each problem fits into one or more category: complete search, divide & conquer, greedy, dynamic programming, ad hoc (i.e., none in particular), approach not listed here namely...

They are in one of the competitive programming systems online, so do try to code them *after* you categorised them. Where they are to be found is on the solution page, not here, because you are expected to have solved it before 'testing' the solution in the online system.

Contents

1 Sharing Chocolate

Chocolate in its many forms is enjoyed by millions of people around the world every day. It is a truly universal candy available in virtually every country around the world.

You find that the only thing better than eating chocolate is to share it with friends. Unfortunately your friends are very picky and have different appetites: some would like more and others less of the chocolate that you offer them. You have found it increasingly difficult to determine whether their demands can be met. It is time to write a program that solves the problem once and for all!

Your chocolate comes as a rectangular bar. The bar consists of same-sized rectangular pieces. To share the chocolate you may break one bar into two pieces along a division between rows or columns of the bar. You or the may then repeatedly break the resulting pieces in the same manner. Each of your friends insists on a getting a single rectangular portion of the chocolate that has a specified number of pieces. You are a little bit insistent as well: you will break up your bar only if all of it can be distributed to your friends, with none left over.

For example, Figure 1 shows one way that a chocolate bar consisting of 3 x 4 pieces can be split into 4 parts that contain 6, 3, 2, and 1 pieces respectively, by breaking it 3 times (This corresponds to the first sample input.)

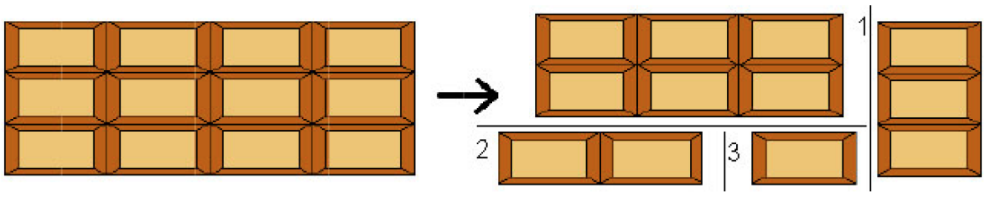


Figure 1: Chocolate sharing example.

Input The input consists of multiple test cases each describing a chocolate bar to share. Each description starts with a line containing a single integer n ($1 \leq n \leq 15$), the number of parts in which the bar is supposed to be split.

This is followed by a line containing two integers x and y ($1 \leq x$, $y \leq 100$), the dimensions of the chocolate bar. The next line contains n positive integers, giving the number of pieces that are supposed to be in each of the n parts.

The input is terminated by a line containing the integer zero.

Output For each test case, first display its case number. Then display whether it is possible to break the chocolate in the desired way: display “`Yes`” if it is possible, and “`No`” otherwise. Follow the format of the sample output.

Sample input

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

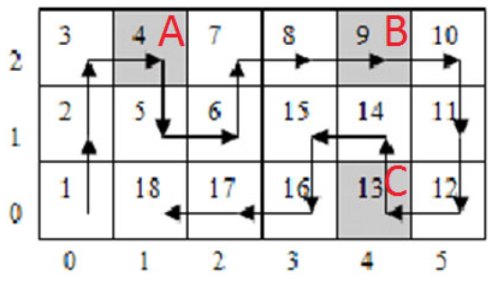
Sample output

```
Case 1: Yes
Case 2: No
```

2 Robots on Ice

Inspired by the ice sculptures in Harbin, the members of the programming team from Arctic University of Robotics and Automata have decided to hold their own ice festival when they return home from the contest. They plan to harvest blocks of ice from a nearby lake when it freezes during the winter. To make it easier to monitor the thickness of the ice, they will lay out a rectangular grid over the surface of the lake and have a lightweight robot travel from square to square to measure ice thickness at each square in the grid. Three locations in the grid are specified as "check-in" points and the robot is supposed to radio a progress report from these points when it is one-fourth, one-half, and three-fourths of the way through its tour of inspection. To avoid unnecessary wear and tear on the surface of the ice, the robot must begin its tour of the grid from the lower left corner, designated in (row,column) coordinates as (0,0), visiting every other grid location exactly once and ending its tour in row 0, column 1. Moreover, if there are multiple tours that the robot can follow, then a different one is to be used each day. The robot is able to move only one square per time step in one of the four compass directions: north, south, east, or west.

You are to design a program that determines how many different tours are possible for a given grid size and a sequence of three check-in points. For example, suppose the lake surface is marked off in a 3 x 6 grid and that the check-in points, in order of visitation, are (2,1), (2,4), and (0,4). Then the robot must start at (0,0) and end at (0,1) after visiting all 18 squares. It must visit location (2,1) on step 4 (= $\lfloor 18/4 \rfloor$), location (2,4) on step 9 (= $\lfloor 18/2 \rfloor$), and location (0,4) on step 13 (= $\lfloor 3 \times 18/4 \rfloor$). There are just two ways to do this (see Figure 2).



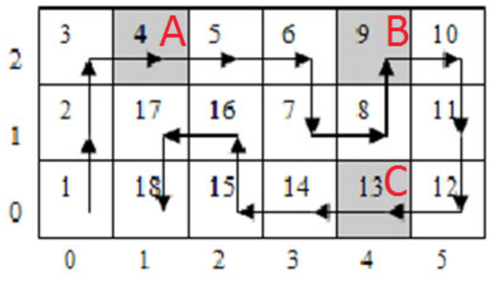


Figure 2: Robot tour example.

Note that when the size of the grid is not divisible by 4, truncated division is used to determine the three check-in times.

Note that some configurations may not permit any valid tours at all. For example, in a 4 x 3 grid with check-in sequence (2,0), (3,2), and (0,2), there is no tour of the grid that begins at (0,0) and ends at (0,1).

Input The input contains several test cases. Each test case begins with a line containing two integers m and n, where $2 \le m, n \le 8$, specifying the number of rows and columns, respectively, in the grid. This is followed by a line containing six integer values r_1, c_1, r_2, c_2, and r_3, c_3, where $0 \le r_i < m$ and $0 \le c_i < n$ for $i = 1, 2, 3$.

Following the last test case is a line containing two zeros.

Output Display the case number, beginning at 1, followed by the number of possible tours that begin at row 0, column 0, end at row 0, column 1, and visit row r_i, column

c_i at time $\lfloor i \times m \times n/4 \rfloor$ for $i = 1, 2, 3$. Follow the format of the sample output.

Sample input

```
3 6
2 1 2 4 0 4
4 3
2 0 3 2 0 2
0 0
```

Sample output

```
Case 1: 2
Case 2: 0
```

3 Consanguine Calculations

Every person's blood has 2 markers called ABO alleles. Each of the markers is represented by one of three letters: A, B, or O. This gives six possible combinations of these alleles that a person can have, each of them resulting in a particular ABO blood type for that person.

Combination	ABO Blood Type
AA	A
AB	AB
AO	A
BB	B
BO	B
OO	O

Likewise, every person has two alleles for the blood Rh factor, represented by the characters + and -. Someone who is "Rh positive" or "Rh+" has at least one + allele, but could have two. Someone who is "Rh negative" always has two – alleles.

The blood type of a person is a combination of ABO blood type and Rh factor. The blood type is written by suffixing the ABO blood type with the + or - representing the Rh factor. Examples include A+, AB-, and O-.

Blood types are inherited: each biological parent donates one ABO allele (randomly chosen from their two) and one Rh factor allele to their child. Therefore 2 ABO alleles and 2 Rh factor alleles of the parents determine the child's blood type. For example, if both parents of a child have blood type A-, then the child could have either type A- or type O- blood. A child of parents with blood types A+ and B+ could have any blood type.

In this problem, you will be given the blood type of either both parents or one parent and a child; you will then determine the (possibly empty) set of blood types that might characterise the child or the other parent. Note: an uppercase letter "Oh" is used in this problem to denote blood types, not a digit (zero).

Input The input consists of multiple test cases. Each test case is on a single line in the format: the blood type of one parent, the blood type of the other parent, and finally the blood type of the child, except that the blood type of one parent or the child will be replaced by a question mark. To improve readability, whitespace may be included anywhere on the line except inside a single blood type specification.

The last test case is followed by a line containing the letters `E`, `N`, and `D` separated by whitespace.

Output For each test case in the input, print the case number (beginning with 1) and the blood type of the parents and the child. If no blood type for a parent is possible, print "`IMPOSSIBLE`". If multiple blood types for parents or child are possible, print all possible values in a comma-separated list enclosed in curly braces. The order of the blood types inside the curly braces does not matter.

The sample output illustrates multiple output formats. Your output format should be similar.

Sample input

```
O+ O- ?
O+ ? O
AB- AB+ ?
AB+ ? O+
E N D
```

Sample output

```
Case 1: O+ O- {O+, O-}
Case 2: O+ {A-, A+, B-, B+, O-, O+} O-
Case 3: AB- AB+ {A+, A-, B+, B-, AB+, AB-}
Case 4: AB+ IMPOSSIBLE O+
```

4 Wire Crossing

Time limit: 2.000 seconds

Moore's Law states that the number of transistors on a chip will double every two years. Amazingly, this law has held true for over half a century. Whenever current technology no longer allowed more growth, researchers have come up with new manufacturing technologies to pack circuits even denser. In the near future, this might mean that chips are constructed in three dimensions instead two. But for this problem, two dimensions will be enough.

A problem common to all two-dimensional hardware design (for example chips, graphics cards, motherboards, and so on) is wire placement. Whenever wires are routed on the hardware, it is problematic if they have to cross each other. When a crossing occurs special gadgets have to be used to allow two electrical wires to pass over each other, and this makes manufacturing more expensive.

Our problem is the following: you are given a hardware design with several wires already in place (all of them straight line segments). You are also given the start and end points for a new wire connection to be added. You will have to determine the minimum number of existing wires that have to be crossed in order to connect the start and end points. This connection need not be a straight line. The only requirement is that it cannot cross at a point where two or more wires already meet or intersect.

Figure 3 shows the first sample input. Eight existing wires form five squares. The start and end points of the new connection are in the leftmost and rightmost squares, respectively. The black dashed line shows that a direct connection would cross four wires, whereas the optimal solution crosses only two wires (the curved blue line).

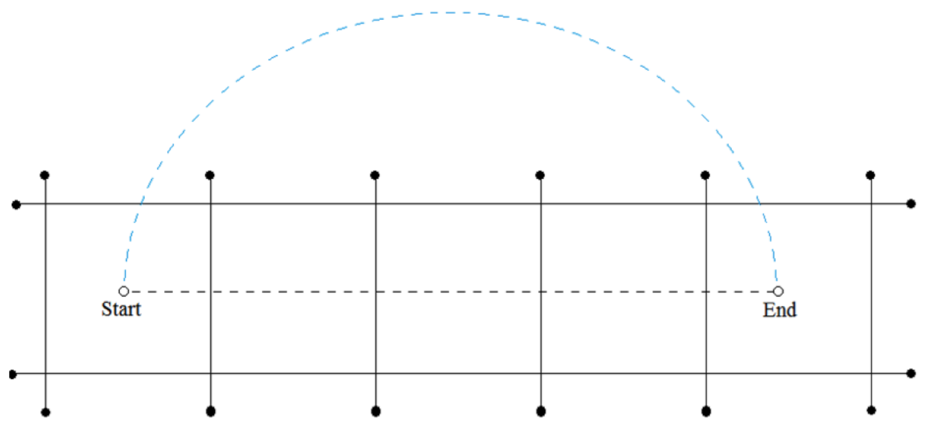


Figure 3: First sample input.

Input The input consists of a single test case. The first line contains five integers m; x_0; y_0; x_1; y_1, which are the number of pre-existing wires ($m \leq 100$) and the start and end points that need to be connected.

This is followed by m lines, each containing four integers x_a, y_a, x_b, y_b describing an existing wire of non-zero length from (x_a, y_a) to (x_b, y_b). The absolute value of each input coordinate is less than 10^5. Each pair of wires has at most one point in common, that is, wires do not overlap. The start and end points for the new wire do not lie on a pre-existing wire.

Output Display the minimum number of wires that have to be crossed to connect the start and end points.

Sample input (example 1)

```
8 3 3 19 3
0 1 22 1
0 5 22 5
1 0 1 6
5 0 5 6
9 0 9 6
13 0 13 6
17 0 17 6
21 0 21 6
```

Sample output (example 1)

```
2
```

Sample input (example 2)

```
1 0 5 10 5
0 0 10 10
```

Sample output (example 2)

```
0
```

5 Word chain

John and Mary are siblings, and are enthusiastic about their courses on foreign languages. Each of the siblings is taking several language courses. When they get home they comment on grammar, vocabulary, culture of the different countries and so on. In one of those conversations they realised some words are common to more than one language, even though the words may have different meanings in the languages. For example, the word “amigo” exists in Portuguese and Spanish and has the same meaning, while “date” is a word that exists in English and French and may have different meanings, since “date” is also a fruit, besides meaning a calendar date. On the other hand, “red” in Spanish is a network, while in English it is a colour.

Thrilled by these findings, the siblings decided to write in a notepad all words in common they could think of, associating each word to a pair of languages. Observant and smart, John proposed a challenge to Mary: given one language to start and one language to finish, write down a sequence of words such that the first word is included in the vocabulary of the start language, and the last word is included in the vocabulary of the finish language. Two adjacent words in the sequence must be in the vocabulary of the same language. For example, if the start language is Portuguese and the finish language is French, Mary could write the sequence “amigo actual date” (Portuguese/Spanish, Spanish/English, English/French).

To John’s surprise, Mary solved the problem rather easily. Annoyed by his sister’s success, he decided to make the problem more difficult: Mary must find a solution in which the sequence has the smallest number of letters in total (not counting spaces between words), and, besides, two consecutive words must not have the same initial letter.

Note that the previous solution is now invalid, as “amigo” and “actual” share the same initial letter. It is possible, however, to find another solution, “amigo red date”, with a total length equal to 12.

John did an extensive research on the Internet and compiled an enormous list of words, and challenged Mary to solve the problem. As there may be more than one solution, he asked her to answer if there is a solution, and in that case to answer the number of letters in the best solution. Can you help Mary?

Input The input contains several test cases. The first line of a test case contains one integer M ($1 \leq M \leq 2000$), representing the total number of words compiled by John. The second line contains two distinct strings O and D, separated by one space, indicating respectively the start language and the finish language.

Each of the next M lines contains three strings I_1, I_2 and P, separated by one space, representing respectively two languages and one word in common between both languages (I_1 and I_2 are always distinct). All strings will have length at least 1 and at most 50, and will be composed of lower case letters only. The same pair of languages may have several words associated to it, but a word P will be never repeated in a test case.

The end of input is indicated by a line containing only one zero.

Output For each test case in the input, your program must print a line with a single integer, the length of the shortest sequence that satisfies John’s restrictions, or the word ‘`impossivel`’ (lowercase, meaning “impossible” in Portuguese) in case it is not possible.

Sample input

```
4
portugues frances
ingles espanhol red
espanhol portugues amigo
frances ingles date
espanhol ingles actual
4
portugues alemao
ingles espanhol red
espanhol portugues amigo
frances ingles date
espanhol ingles actual
6
portugues frances
ingles espanhol red
espanhol portugues amigo
frances ingles date
frances espanhol la
portugues ingles a
espanhol ingles actual
0
```

Sample output

```
12
impossivel
5
```

6 Shopping for operas

For some reason, Ndumiso loves to collect and watch operas on DVD. He can find and order all the operas he wants from Kalahari.net, and they will even deliver them right to her door, but he can usually find a better price at one of his favourite stores. However, with the cost of petrol nowadays, it is hard to tell whether or not one would actually save money by driving to the stores to purchase the DVDs.

Ndumiso would like to buy some operas today. For each of the operas he wants, he knows exactly one store that is selling it for a lower cost than the Kalahari.net price. He would like to know whether it would actually be worth it to go out and buy the operas from the stores. Ndumiso only knows the road system connecting his favourite stores, and will only use those roads to get around. He knows at least one route, if only an indirect one, to every store.

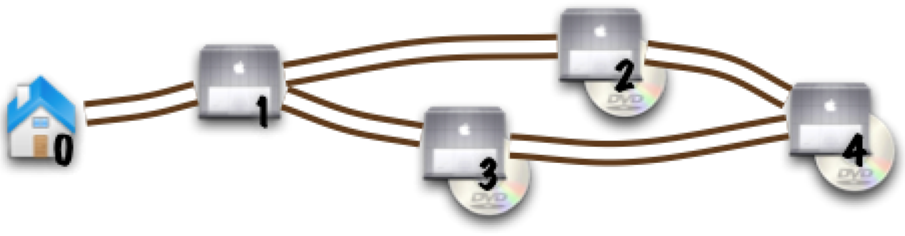


Figure 4: An example diagram of Ndumiso's house, his favourite stores, and the roads connecting them

In his shopping trip, Ndumiso begins at his house, drives from store to store in any order to purchase his operas, then drives back to his house. For any particular opera, he can opt not to drive to the store to buy it, since he can just order it from Kalahari.net.

For convenience, Ndumiso assigned his house the integer 0, and numbered each of his favourite stores with integers starting at 1. You are given a description of the road system and the exact amount it would cost for Ndumiso to drive each road. For each opera Ndumiso wants, you are given the number of the store it is available at, and the amount he would save if he bought that particular opera at that store. Your task is to determine the greatest amount of money Ndumiso can save by making the shopping trip.

Input The first line of input contains a single number indicating the number of scenarios to process. A blank line precedes each scenario.

Each scenario begins with line containing two numbers: N ($1 \leq N \leq 50$), the number of stores, and M ($1 \leq M \leq 1000$), the number of roads. The following M lines each contain a description of a road. Each road is described by two integers indicating the house or stores it connects, and a real number with two decimal digits indicating the cost in Rand to drive that road. All roads are two-way.

The next line in the scenario contains a number P ($1 \leq P \leq 12$), the number of opera DVDs Ndumiso wants to buy. For each of the P operas, a line follows containing an integer indicating the store number at which the opera is available, and a real number with two decimal digits indicating the difference between the Kalahari.net price and the price at that store in Rand.

Output For each scenario in the input, write one line of output indicating the largest amount of money, in Rands and cents, that Ndumiso can save by making his shopping trip. Follow the format of the sample output; there should always be two digits after the decimal point to indicate the number of cents. If Ndumiso cannot save any money by going to the stores, output a single line saying "Don't leave the house".

Sample input

```
2

4 5
0 1 1.00
1 2 3.00
1 3 2.00
2 4 4.00
3 4 3.25
3
2 1.50
3 7.00
4 9.00

1 1
0 1 1.50
1
1 2.99
```

Sample output

```
Ndumiso can save ZAR 3.50
Don't leave the house
```

Solution

This page doesn't give you the solution to the actual problem, only the category and which UVa problem it is. You still have to solve and code it. You can test your solution at `http://uva.onlinejudge.org/`.

Problem 1: Sharing Chocolate. This is UVa problem 1099, from the World Finals in Harbin, 2010. Solvable using DP; an explanation is available in §8.3.6 of CP3.

Problem 2: Robots on Ice. This is UVa problem 1098, from the World Finals in Harbin, 2010. Solvable using Meet in the Middle; an explanation is available in §8.2.2 and §8.2.4 of CP3.

Problem 3: Consanguine Calculations. This is UVa problem 1061, from the World Finals in Tokyo, 2007. Solvable by trying all eight possible blood + Rh types.

Problem 4: Wire Crossing. This is UVa problem 1xxx, and was also Problem M in the World Finals in Ekaterinburg, 2014. Solvable using Geometry + Shortest path; an explanation is available at `http://blog.brucemerry.org.za/2014/06/icpc-problem-m-wires.html`

Problem 5: Word chain. This is UVa problem "babel", 11492. Solvable using graph modelling, then Single-Source Shortest Path; see summary on p154 of CP3.

Problem 6: Shopping for operas. This is essentially UVa problem "Shopping trip", 11284; if you try it in UVa, use the pattern "`Daniel can save $3.50`" in the output instead of "`Ndumiso can save ZAR 3.50`". Solvable using DP (it's a traveling salesman problem, adjusted a little for the option to go home early; see also §8.4.2 and §9.2 of CP3).