Problem Solving Strategies – set of 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 Charming canines

Input: Standard Input; Output: Standard Output; Time Limit: 1 Second

Once upon a time, there lived a lion called Luchu Bandor. When Luchu was young, he was given cute little poodle as pet, called Bunty Mona. Luchu had grown up and was feeling uncomfortable taking Bunty to public places along with him, as everyone would stare at them all the while. At one point, Luchu could not stand it anymore and started looking for a new dog in the dog kennel and training school, as he still did like having a dog as pet. Every day Luchu would wait for the morning drill to start. From there he could see each all dogs doing their routine drill. Now Luchu was looking for the tallest dog that would be shorter than he; he would also like to consider a dog a little taller than he. But a dog of his same height will never be on his list. Every morning Luchu picks up a line of dogs and finds the best two according to his set criterion. His job has been made easy by the fact that the dogs in each line are ordered by their height, the shortest one is in the front and the tallest one is at the back. Your task is to help Luchu on one particular day to find two dogs: the tallest one shorter than he and the shortest one taller than he.

There will be only one set of input for this problem. The first line of input gives you a number N ($1 \leq N \leq 50000$), the number of dogs on the line. In the next line you would have N integers (in the range 1 to 2^{31}-1) giving the heights of the N dogs. There would be a single space after every number. You can assume that the dogs are ordered in non-decreasing order of their heights. In the next line you would have an integer Q ($1 \leq Q \leq 25000$) giving the number of queries. Then in the next line Q queries will follow. Then you would have Q numbers giving the height of Luchu! Don't worry, Luchu is from the land where people can have 3 birth dates; Q heights for a dog will make no difference here. The Q numbers are listed on a line and their range from 1 to 2^{31}-1, and as before you would find a single space after every query number. The query numbers are not supposed to come in any particular order.

For each query height, print two numbers in one line. The first one would be the height of the tallest dog that is shorter than Luchu, and the next number would be the height of the shortest dog that is taller than he. These two numbers are to be separated by a single space. Whenever it is impossible to find any of these two heights, replace that height with an uppercase 'X'.

Sample input

```
4
1 4 5 7
4
4 6 8 10
```

Sample output

```
1 5
5 7
7 X
7 X
```

/*Note: You should at least use scanf() and printf() to take input and produce output for this problem. cin and cout is too slow for this problem to get it within time limit. Sorry for this but otherwise it becomes impossible for us to set a reasonable time limit.*/

2 Work reduction

Time limit: 3.000 seconds

Paperwork is beginning to pile up on your desk, and tensions at the workplace are starting to mount. Your boss has threatened to fire you if you don't make any progress by the end of the day. You currently have N units of paperwork on your desk, and your boss demands that you have exactly M units of paperwork left by the end of the day.

The only hope for you now is to hire help. There are various agencies which offer paperwork reduction plans:
For \$A they will reduce your paperwork by one unit.
For \$B they will reduce your entire paperwork by half (rounding down when necessary).
Note that work can never be reduced to less than 0.

Your task now is to produce a sorted table of agency names and their respective minimum costs to solve your workload problem.

The first line of input consists of a single positive integer representing the number of cases to follow. Each case begins with three positive integers separated by spaces: N – your starting workload, M – your target workload, and L – the number of work reduction agencies available to you, $1 \leq M \leq N \leq 100000$, $1 \leq L \leq 100$. The next L lines have the format "`[agency name]:A,B`", where A and B are the rates as described above for the given agency ($0 \leq A, B \leq 10000$). The length of the agency name will be between 1 and 16, and will consist only of capital letters. Agency names will be unique.

For each test case, print "Case X", with X being the case number, on a single line, followed by the table of agency names and their respective minimum costs, sorted in non-decreasing order of minimum costs. Sort job agencies with identical minimum costs in alphabetical order by agency name. For each line of the table, print out the agency name, followed by a space, followed by the minimum required cost for that agency to solve your problem.

Sample input

```
2
100 5 3
A:1,10
B:2,5
C:3,1
1123 1122 5
B:50,300
A:1,1000
C:10,10
D:1,50
E:0,0
```

Sample output

```
Case 1
C 7
B 22
A 37
Case 2
E 0
A 1
D 1
C 10
B 50
```

3 Trainsorting

Time limit: 1.000 seconds

Erin is an engineer. She drives trains. She also arranges the cars within each train. She prefers to put the cars in decreasing order of weight, with the heaviest car at the front of the train. Unfortunately, sorting train cars is not easy. One cannot simply pick up a car and place it somewhere else. It is impractical to insert a car within an existing train. A car may only be added to the beginning and end of the train. Cars arrive at the train station in a predetermined order. When each car arrives, Erin can add it to the beginning or end of her train, or refuse to add it at all. The resulting train should be as long as possible, but the cars within it must be ordered by weight. Given the weights of the cars in the order in which they arrive, what is the longest train that Erin can make?

The first line is the number of test cases to follow. The test cases follow, one after another; the format of each test case is the following: The first line contains an integer $0 \leq n \leq 2000$, the number of cars. Each of the following n lines contains a non-negative integer giving the weight of a car. No two cars have the same weight.

Output a single integer giving the number of cars in the longest train that can be made with the given restrictions.

Sample input

```
1
3
1
2
3
```

Sample output

```
3
```

4 Wine trading in Gergovia

Time limit: 3.000 seconds

As you may know from the comic “Asterix and the Chieftain’s Shield”, Gergovia consists of one street, and every inhabitant of the city is a wine salesman. You wonder how this economy works? Simple enough: everyone buys wine from other inhabitants of the city. Every day each inhabitant decides how much wine he wants to buy or sell. Interestingly, demand and supply is always the same, so that each inhabitant gets what he wants.

There is one problem, however: Transporting wine from one house to another results in work. Since all wines are equally good, the inhabitants of Gergovia don’t care which persons they are doing trade with, they are only interested in selling or buying a specific amount of wine. They are clever enough to figure out a way of trading so that the overall amount of work needed for transports is minimized.

In this problem you are asked to reconstruct the trading during one day in Gergovia. For simplicity we will assume that the houses are built along a straight line with equal distance between adjacent houses. Transporting one bottle of wine from one house to an adjacent house results in one unit of work.

Input Specification. The input consists of several test cases. Each test case starts with the number of inhabitants n where $2 \leq n \leq 100000$. The following line contains n integers a_i ($-1000 \leq a_i \leq 1000$). If $a_i \geq 0$, it means that the inhabitant living in the ith house wants to buy a_i bottles of wine, otherwise if $a_i < 0$, he wants to sell $-a_i$ bottles of wine. You may assume that the numbers a_i sum up to 0. The last test case is followed by a line containing 0.

Output Specification. For each test case print the minimum amount of work units needed so that every inhabitant has his demand fulfilled. You may assume that this number fits into a signed 64-bit integer (in C/C++ you can use the data type “long long”, in JAVA the data type “long”).

Sample input

```
5
5 -4 1 -3 1
6
-1000 -1000 -1000 1000 1000 1000
0
```

Sample output

```
9
9000
```

5 The jackpot

Note: Time Limit: 1 second; Memory Limit: 1 Mb

As Manuel wants to get rich fast and without too much work, he decided to make a career in gambling. Initially, he plans to study the gains and losses of players, so that, he can identify patterns of consecutive wins and elaborate a win-win strategy. But Manuel, as smart as he thinks he is, does not know how to program computers. So he hired you to write programs that will assist him in elaborating his strategy.

Your first task is to write a program that identifies the maximum possible gain out of a sequence of bets. A bet is an amount of money and is either winning (and this is recorded as a positive value), or losing (and this is recorded as a negative value).

The input set consists of a positive number $N \leq 10000$, that gives the length of the sequence, followed by N integers. Each bet is an integer greater than 0 and less than 1000. The input is terminated with $N = 0$.

For each given input set, the output will echo a line with the corresponding solution. If the sequence shows no possibility to win money, then the output is the message "Losing streak."

Sample input

```
5
12 -4
-10 4
9
3
-2 -1 -2
0
```

Sample output

```
The maximum winning streak is 13.
Losing streak.
```

6 Durban Prawns

Input: standard input; Output: standard output; Time Limit: 7 seconds; Memory Limit: 32 MB

Durban has a lovely subtropical climate. What unsuspecting visitors may not realise, is that cockroaches (of the 7cm long and 1.3cm thick type) like that weather too! Durban can't really get rid of them, and Durbanites call them affectionately "Durban prawns". However, recently things have gotten a bit out of control, and some extermination programme has to be put in place. As keeping up appearances is important, the eThekwini Municipality decides that only the streets have to be clean from roaches. They have tendered the cleanup to the African Catastrophe Management (ACM) company.

The African Catastrophe Management has scouted for roach nests, having observed that they always set up their nests on street intersections, and determined that the only viable method to extinguish them is to use gas bombs. However, gas bombs are not only lethal to the roaches, but also for humans, so the buildings in the area of the gas bomb have to be evacuated in advance. To optimize the effect of exterminating the prawns, and minimize evacuations, the point of setting of the gas bomb must be chosen very carefully. African Catastrophe Management uses 'smart gas', for when fired, the gas spreads in a rectangular fashion through the streets. The strength of a gas bomb is given by a number d which specifies the rectangular 'radius' of the gas diffusion area. Take, for example, the section of Durban shown in Fig. 1, and a stylised version of that (on the right), which shows what happens when a bomb with $d = 1$ explodes. Assume all of Durban has its streets in a rectangular grid.

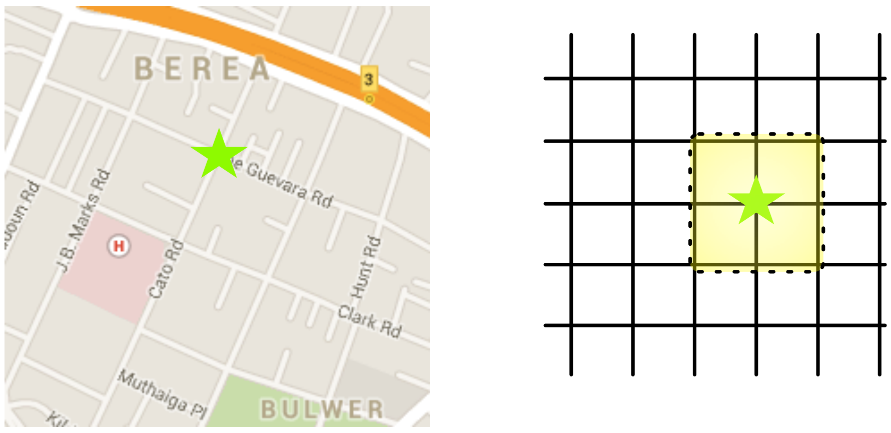

Figure 1: left: small section of the map of Durban, with the gas bomb (green star) on the corner of Cato road and Che Guevara road; right: suitable abstraction with the bomb (green star) and a diffusion area when $d = 1$ (yellow rectangle).

The area of interest consists of a discrete grid of 1025×1025 fields. The cockroach exterminator scouts have given a detailed report on where roach populations of different sizes have built their nests. You are given a gas bomb with strength d and your task is to find an explosion location for this gas bomb which extinguishes the largest number of roaches. The best position is determined by the following criteria:

- The sum of all roach population sizes within the diffusion area of the gas bomb (given by d) is maximal.

- If there is more than one of these best positions then the location with the "minimal" position will be chosen. Positions are ordered first by their x coordinate and second by their y coordinate.

Formally, given a location (x_1, y_1) on the grid, a point (x_2, y_2) is within the diffusion area of a gas bomb with strength d if the following equation holds:

$$max(abs(x_2 - x_1), abs(y_2 - y_1)) \leq d \tag{1}$$

Regarding the input, the first line contains the number of scenarios in the input. For each scenario the first line contains the strength d of the gas bomb in the scenario ($1 \leq d \leq 50$). The second line contains the number n ($1 \leq n \leq 20000$) of cockroach populations. Then for every roach population follows a line containing three integers separated by spaces for the position (x, y) and 'size' i of the population ($1 \leq i \leq 255$). It is guaranteed that position coordinates are valid (i.e., in the range between 0 and 1024) and no position is given more than once.

Output. For every problem print a line containing the x and y coordinate of the chosen location for the gas bomb, followed by the sum of the roach population sizes which will be extinguished. The three numbers must be separated by a space.

Sample input

```
1
1
2
4 4 10
6 6 20
```

Sample output

```
5 5 30
```

7 Mobile phone coverage

Time limit: 3.000 seconds

The mobile phone company Advanced Cellular, Mobile, and Internet-Connected Phone Corporation (ACMICPC) is planning to set up a collection of antennas for mobile phones in a city called Maxnorm. ACMICPC has several collections for locations of antennas as their candidate plans, and now they want to know which collection is the best choice.

For this purpose, they want to develop a computer program to find the coverage of a collection of antenna locations. Each antenna A_i has power r_i, corresponding to "radius". Usually, the coverage region of the antenna may be modeled as a disk centered at the location of the antenna (x_i, y_i) with radius r_i. However, in this city Maxnorm such a coverage region becomes the square $[x_i - r_i, x_i + r_i] \times [y_i - r_i, y_i + r_i]$. In other words, the distance between two points (x_p, y_p) and (x_q, y_q) is measured by the max norm $\max\{|x_p - x_q|, |y_p - y_q|\}$, or, the L_∞ norm, in this city Maxnorm instead of the ordinary Euclidean norm $\sqrt{(x_p - x_q)^2 + (y_p - y_q)^2}$.

As an example, consider the following collection of 3 antennas, depicted in Fig. 2.

```
4.0 4.0 3.0
5.0 6.0 3.0
5.5 4.5 1.0
```

where the i-th row represents x_i, y_i, r_i such that (x_i, y_i) is the position of the i-th antenna and r_i is its power. The area of regions of points covered by at least one antenna is 52.00 in this case.

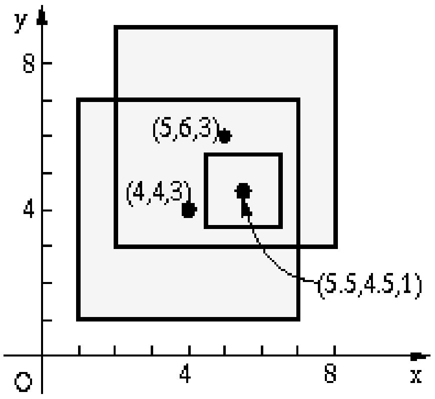

Figure 2: Illustration for the antennas and coverage.

Write a program that finds the area of coverage by a given collection of antenna locations.

The input contains multiple data sets, each representing a collection of antenna locations. A data set is given in the following format.

$$\begin{array}{lll} n & & \\ x_1 & y_1 & r_1 \\ x_2 & y_2 & r_2 \\ \ldots & & \\ x_n & y_n & r_n \end{array}$$

The first integer n is the number of antennas, such that $2 \leq n \leq 100$. The coordinate of the i-th antenna is given by (x_i, y_i), and its power is r_i. x_i, y_i and r_i are fractional numbers between 0 and 200 inclusive. The end of the input is indicated by a data set with 0 as the value of n.

For each data set, your program should output its sequence number (1 for the first data set, 2 for the second, etc.) and the area of the coverage region. The area should be printed with two digits to the right of the decimal point, after rounding it to two decimal places. The sequence number and the area should be printed on the same line with no spaces at the beginning and end of the line. The two numbers should be separated by a space.

Sample input

```
3
4.0 4.0 3.0
5.0 6.0 3.0
5.5 4.5 1.0
2
3.0 3.0 3.0
1.5 1.5 1.0
0
```

Sample output

```
1 52.00
2 36.00
```

8 Blowing fuses

Maybe you are familiar with the following situation. You have plugged in several electrical devices, such as a toaster, refrigerator, microwave oven, computers, a stereo, etc, and have them all running. But at the moment when you turn on the TV, the fuse blows, since the power drawn from all the machines is greater than the capacity of the fuse. Of course this is a great safety feature, avoiding that houses burn down too often due to fires ignited by overheating wires. But it is also annoying to walk down to the basement (or some other inconvenient place) to replace to fuse or switch it back on.

What one would like to have is a program that checks *before* turning on an electrical device whether the combined power drawn by all running devices exceeds the fuses capacity (and it blows), or whether it is safe to turn it on.

The input consists of several test cases. Each test case describes a set of electrical devices and gives a sequence of turn on/off operations for these devices. The first line of each test case contains three integers n, m and c, where n is the number of devices ($n \leq 20$), m the number of operations performed on these devices and c is the capacity of the fuse (in Amperes). The following n lines contain one positive integer c_i each, the consumption (in Amperes) of the i-th device. This is followed by m lines also containing one integer each, between 1 and n inclusive. They describe a sequence of turn on/turn off operations performed on the devices. For every number, the state of that particular devices is toggled, i.e., if it is currently running, it is turned off, and if it is currently turned off, it will by switched on. At the beginning all devices are turned off. The input will be terminated by a test case starting with $n = m = c = 0$. This test case should not be processed.

For each test case, first output the number of the test case. Then output whether the fuse was blown during the operation sequence. The fuse will be blown if the sum of the power consumptions c_i of turned on devices at some point exceeds the capacity of the fuse c. If the fuse is not blown, output the maximal power consumption by turned on devices that occurred during the sequence. Output a blank line after each test case.

Sample input

```
2 2 10
5
7
1
2
3 6 10
2
5
7
2
1
2
3
1
3
0 0 0
```

Sample output

```
Sequence 1
Fuse was blown.

Sequence 2
Fuse was not blown.
Maximal power consumption was 9 amperes.
```

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: Charming canines. This is essentially UVa problem 10611; I changed the storyline, but all the variables, input etc. are exactly the same. Solvable using Divide & Conquer (binary search).

Problem 2: Work reduction. This is UVa problem 10670. Solvable using Greedy.

Problem 3: Trainsorting. This is UVa problem 11456. Solvable using Dynamic Programming.

Problem 4: Wine trading in Gergovia. This is UVa problem 11054, and also used in the 2006/2007 ACM ICPC Local Contest at the University of Ulm. Solvable using Greedy.

Problem 5: The jackpot. This is UVa problem 10684. Solvable using Dynamic Programming (maximum 1-D sum/maximum consecutive subsequence)

Problem 6: Durban prawns. This is essentially UVa problem 10360, which is from the TUD Programming Contest; I changed the storyline, but all the variables, input etc. are exactly the same. Solvable using Complete Search.

Problem 7: Mobile Phone Coverage. This is UVa problem 688. Solvable using Complete Search + Geometry.

Problem 8: Blowing fuses. This is UVa problem 661. Solvable using an ad hoc algorithm (simulation).