Nordic Collegiate Programming Contest
NCPC 2007

October 6th, 2007

The Problemset

Do not look at the problems before the contest has started.

Problem A

Phone List

Given a list of phone numbers, determine if it is consistent in the sense that no number is the prefix of another. Let's say the phone catalogue listed these numbers:

- Emergency 911

- Alice 97 625 999

- Bob 91 12 54 26

In this case, it's not possible to call Bob, because the central would direct your call to the emergency line as soon as you had dialled the first three digits of Bob's phone number. So this list would not be consistent.

Input specifications

The first line of input gives a single integer, $1 \le t \le 40$, the number of test cases. Each test case starts with n, the number of phone numbers, on a separate line, $1 \le n \le 10000$. Then follows n lines with one unique phone number on each line. A phone number is a sequence of at most ten digits.

Output specifications

For each test case, output "YES" if the list is consistent, or "NO" otherwise.

Sample input	Output for sample input
2	NO
3	YES
911	
97625999	
91125426	
5	
113	
12340	
123440	
12345	
98346	

Problem B

Cuckoo Hashing

One of the most fundamental data structure problems is the dictionary problem: given a set D of words you want to be able to quickly determine if any given query string q is present in the dictionary D or not. Hashing is a well-known solution for the problem. The idea is to create a function $h : \Sigma^* \to [0..n-1]$ from all strings to the integer range $0, 1, .., n-1$, i.e. you describe a fast deterministic program which takes a string as input and outputs an integer between 0 and $n-1$. Next you allocate an empty hash table T of size n and for each word w in D, you set $T[h(w)] = w$. Thus, given a query string q, you only need to calculate $h(q)$ and see if $T[h(q)]$ equals q, to determine if q is in the dictionary. Seems simple enough, but aren't we forgetting something? Of course, what if two words in D map to the same location in the table? This phenomenon, called collision, happens fairly often (remember the Birthday paradox: in a class of 24 pupils there is more than 50% chance that two of them share birthday). On average you will only be able to put roughly $\sqrt{n}$-sized dictionaries into the table without getting collisions, quite poor space usage!

A stronger variant is Cuckoo Hashing[1]. The idea is to use two hash functions h_1 and h_2. Thus each string maps to two positions in the table. A query string q is now handled as follows: you compute both $h_1(q)$ and $h_2(q)$, and if $T[h_1(q)] = q$, *or* $T[h_2(q)] = q$, you conclude that q is in D. The name "Cuckoo Hashing" stems from the process of creating the table. Initially you have an empty table. You iterate over the words d in D, and insert them one by one. If $T[h_1(d)]$ is free, you set $T[h_1(d)] = d$. Otherwise if $T[h_2(d)]$ is free, you set $T[h_2(d)] = d$. If both are occupied however, just like the cuckoo with other birds' eggs, you evict the word r in $T[h_1(d)]$ and set $T[h_1(d)] = d$. Next you put r back into the table in its alternative place (and if that entry was already occupied you evict that word and move it to its alternative place, and so on). Of course, we may end up in an infinite loop here, in which case we need to rebuild the table with other choices of hash functions. The good news is that this will not happen with great probability even if D contains up to $n/2$ words!

Input specifications

On the first line of input is a single positive integer $1 \le t \le 50$ specifying the number of test cases to follow. Each test case begins with two positive integers $1 \le m \le n \le 10000$ on a line of itself, m telling the number of words in the dictionary and n the size of the hash table in the test case. Next follow m lines of which the i:th describes the i:th word d_i in the dictionary D by two non-negative integers $h_1(d_i)$ and $h_2(d_i)$ less than n giving

---

[1]Cuckoo Hashing was suggested by the danes R. Pagh and F. F. Rödler in 2001

the two hash function values of the word d_i. The two values may be identical.

Output specifications

For each test case there should be exactly one line of output either containing the string "`successful hashing`" if it is possible to insert all words in the given order into the table, or the string "`rehash necessary`" if it is impossible.

Sample input	Output for sample input
2	successful hashing
3 3	rehash necessary
0 1	
1 2	
2 0	
5 6	
2 3	
3 1	
1 2	
5 1	
2 5	

Problem C

Optimal Parking

When shopping on Long Street, Michael usually parks his car at some random location, and then walks to the stores he needs. Can you help Michael choose a place to park which minimises the distance he needs to walk on his shopping round?

Long Street is a straight line, where all positions are integer. You pay for parking in a specific slot, which is an integer position on Long Street. Michael does not want to pay for more than one parking though. He is very strong, and does not mind carrying all the bags around.

Input specifications

The first line of input gives the number of test cases, $1 \leq t \leq 100$. There are two lines for each test case. The first gives the number of stores Michael wants to visit, $1 \leq n \leq 20$, and the second gives their n integer positions on Long Street, $0 \leq x_i \leq 99$.

Output specifications

Output for each test case a line with the minimal distance Michael must walk given optimal parking.

| Sample input	Output for sample input
2	152
4	70
24 13 89 37	
6	
7 30 41 14 39 42	

Problem D

Copying DNA

Evolution is a seemingly random process which works in a way which resembles certain approaches we use to get approximate solutions to hard combinatorial problems. You are now to do something completely different.

Given a DNA string S from the alphabet {A,C,G,T}, find the minimal number of copy operations needed to create another string T. You may reverse the strings you copy, and copy both from S and the pieces of your partial T. You may put these pieces together at any time. You may only copy contiguous parts of your partial T, and all copied strings must be used in your final T. Example: From $S =$ "ACTG" create $T =$ "GTACTATTATA"
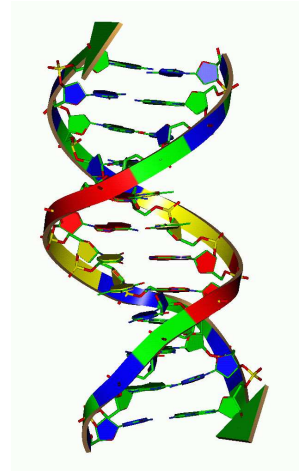
1. Get GT......... by copying and reversing "TG" from S.
2. Get GTAC....... by copying "AC" from S.
3. Get GTAC...TA.. by copying "TA" from the partial T.
4. Get GTAC...TAAT by copying and reversing "TA" from the partial T.
5. Get GTACAATTAAT by copying "AAT" from the partial T.

Input specifications

The first line of input gives a single integer, $1 \le t \le 100$, the number of test cases. Then follow, for each test case, a line with the string S of length $1 \le m \le 18$, and a line with the string T of length $1 \le n \le 18$.

Output specifications

Output for each test case the number of copy operations needed to create T from S, or "impossible" if it cannot be done.

Sample input

```
5
ACGT
GTAC
A
C
ACGT
TGCA
ACGT
TCGATCGA
A
AAAAAAAAAAAAAAAAAA
```

Output for sample input

```
2
impossible
1
4
6
```

Problem E

Circle of Debt

The three friends Alice, Bob, and Cynthia always seem to get in situations where there are debts to be cleared among themselves. Of course, this is the "price" of hanging out a lot: it only takes a few resturant visits, movies, and drink rounds to get an unsettled balance. So when they meet as usual every Friday afternoon they begin their evening by clearing last week's debts. To satisfy their mathematically inclined minds they prefer clearing their debts using as little money transaction as possible, i.e. by exchanging as few bank notes and coins as necessary. To their surprise, this can sometimes by harder than it sounds. Suppose that Alice owes Bob 10 crowns and this is the three friends' only uncleared debt, and Alice has a 50 crown note but nothing smaller, Bob has three 10 crown coins and ten 1 crown coins, and Cynthia has three 20 crown notes. The best way to clear the debt is for Alice to give her 50 crown note to Cynthia, Cynthia to give two 20 crown notes to Alice and one to Bob, and Bob to give one 10 crown coin to Cynthia, involving a total of only five notes/coins changing owners. Compare this to the straight-forward solution of Alice giving her 50 crown note to Bob and getting Bob's three 10 crown notes and all his 1 crown coins for a total of fourteen notes/coins being exchanged!

Input specifications

On the first line of input is a single positive integer, $1 \leq t \leq 50$, specifying the number of test cases to follow. Each test case begins with three integers $ab, bc, ca \leq 1000$ on a line of itself. ab is the amount Alice owes Bob (negative if it is Bob who owes Alice money), bc the amount Bob owes Cynthia (negative if it is Cynthia who is in debt to Bob), and ca the amount Cynthia owes Alice (negative if it is Alice who owes Cynthia).

Next follow three lines each with six non-negative integers $a_{100}, a_{50}, a_{20}, a_{10}, a_5, a_1$, $b_{100}, \ldots, b_1$, and $c_{100}, \ldots c_1$, respectively, where a_{100} is the number of 100 crown notes Alice got, a_{50} is the number of her 50 crown notes, and so on. Likewise, $b_{100}, \ldots, b_1$ is the amount of notes/coins of different value Bob got, and $c_{100}, \ldots, c_1$ describes Cynthia's money. Each of them has at most 30 coins (i.e. $a_{10} + a_5 + a_1$, $b_{10} + b_5 + b_1$, and $c_{10} + c_5 + c_1$ are all less than or equal to 30) and the total amount of all their money together (Alice's plus Bob's plus Cynthia's) is always less than 1000 crowns.

Output specifications

For each test case there should be one line of output containing the minimum number of bank notes and coins needed to settle the balance. If it is not possible at all, output the string "impossible".

Sample input

```
3
10 0 0
0 1 0 0 0 0
0 0 0 3 0 10
0 0 3 0 0 0
-10 -10 -10
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
-10 10 10
3 0 0 0 2 0
0 2 0 0 0 1
0 0 1 1 0 3
```

Output for sample input

```
5
0
impossible
```

Problem F

Full Tank?

After going through the receipts from your car trip through Europe this summer, you realised that the gas prices varied between the cities you visited. Maybe you could have saved some money if you were a bit more clever about where you filled your fuel?

To help other tourists (and save money yourself next time), you want to write a program for finding the cheapest way to travel between cities, filling your tank on the way. We assume that all cars use one unit of fuel per unit of distance, and start with an empty gas tank.

Input specifications

The first line of input gives $1 \le n \le 1000$ and $0 \le m \le 10000$, the number of cities and roads. Then follows a line with n integers $1 \le p_i \le 100$, where p_i is the fuel price in the ith city. Then follow m lines with three integers $0 \le u, v < n$ and $1 \le d \le 100$, telling that there is a road between u and v with length d. Then comes a line with the number $1 \le q \le 100$, giving the number of queries, and q lines with three integers $1 \le c \le 100$, s and e, where c is the fuel capacity of the vehicle, s is the starting city, and e is the goal.

Output specifications

For each query, output the price of the cheapest trip from s to e using a car with the given capacity, or "`impossible`" if there is no way of getting from s to e with the given car.

Sample input

```
5 5
10 10 20 12 13
0 1 9
0 2 8
1 2 1
1 3 11
2 3 7
2
10 0 3
20 1 4
```

Output for sample input

```
170
impossible
```

Problem G

Nested Dolls

Dilworth is the world's most prominent collector of Russian nested dolls: he literally has thousands of them! You know, the wooden hollow dolls of different sizes of which the smallest doll is contained in the second smallest, and this doll is in turn contained in the next one and so forth. One day he wonders if there is another way of nesting them so he will end up with fewer nested dolls? After all, that would make his collection even more magnificent! He unpacks each nested doll and measures the width and height of each contained doll. A doll with width w_1 and height h_1 will fit in another doll of width w_2 and height h_2 if and only if $w_1 < w_2$ and $h_1 < h_2$. Can you help him calculate the smallest number of nested dolls possible to assemble from his massive list of measurements?

Input specifications

On the first line of input is a single positive integer $1 \le t \le 20$ specifying the number of test cases to follow. Each test case begins with a positive integer $1 \le m \le 20000$ on a line of itself telling the number of dolls in the test case. Next follow $2m$ positive integers $w_1, h_1, w_2, h_2, \ldots, w_m, h_m$, where w_i is the width and h_i is the height of doll number i. $1 \le w_i, h_i \le 10000$ for all i.

Output specifications

For each test case there should be one line of output containing the minimum number of nested dolls possible.

Sample input	Output for sample input
4	1
3	2
20 30 40 50 30 40	3
4	2
20 30 10 10 30 20 40 50	
3	
10 30 20 20 30 10	
4	
10 10 20 30 40 50 39 51	

Problem H

Shopaholic

Lindsay is a shopaholic. Whenever there is a discount of the kind where you can buy three items and only pay for two, she goes completely mad and feels a need to buy all items in the store. You have given up on curing her for this disease, but try to limit its effect on her wallet.

You have realized that the stores coming with these offers are quite selective when it comes to which items you get for free; it is always the cheapest ones. As an example, when your friend comes to the counter with seven items, costing 400, 350, 300, 250, 200, 150, and 100 dollars, she will have to pay 1500 dollars. In this case she got a discount of 250 dollars. You realize that if she goes to the counter three times, she might get a bigger discount. E.g. if she goes with the items that costs 400, 300 and 250, she will get a discount of 250 the first round. The next round she brings the item that costs 150 giving no extra discount, but the third round she takes the last items that costs 350, 200 and 100 giving a discount of an additional 100 dollars, adding up to a total discount of 350.

Your job is to find the maximum discount Lindsay can get.

Input specifications

The first line of input gives the number of test scenarios, $1 \le t \le 20$. Each scenario consists of two lines of input. The first gives the number of items Lindsay is buying, $1 \le n \le 20000$. The next line gives the prices of these items, $1 \le p_i \le 20000$.

Output specifications

For each scenario, output one line giving the maximum discount Lindsay can get by selectively choosing which items she brings to the counter at the same time.

Sample input

```
1
6
400 100 200 350 300 250
```

Output for sample input

```
400
```

Problem I

Moogle

You got the original idea of making map software, called Moogle Maps, for the new cool Maple mPhone. It will even be capable of indicating the location of a house address like "Main Street 13". However, since the mPhone has limited storage capacity, you need to reduce the data amount. You don't want to store the exact location of every single house number. Instead only a subset of the house numbers will be stored exactly, and the others will be linearly interpolated. So you want to select house numbers that will minimise the average interpolation error, given how many house locations you have capacity to store. We view the street as a straight line, and you will always store the first and the last house location.

Given that you've stored the locations x_i and x_j for the houses with numbers i and j respectively, but no other house in between, the interpolated value for a house with number k with $i < k < j$ is $x_i + (x_j - x_i) \cdot \frac{k-i}{j-i}$.

Input specifications

The first line of input gives a single integer, $1 \le t \le 50$, the number of test cases.

For each test case, there are two lines. The first contains $2 \le h \le 200$ and $2 \le c \le h$, where h is the number of houses in the street and c is the number of house locations that can be stored. The second contains h integers in increasing order giving the location of the h houses. Each location is in the interval $[0, 1000000]$.

Output specifications

For each test case, output the average interpolation error over all the h houses for the optimal selection of c house locations to store. The output should be given with four decimal places, but we will accept inaccuracies of up to ± 0.001.

Sample input

```
2
4 3
0 9 20 40
10 4
0 10 19 30 40 90 140 190 202 210
```

Output for sample input

```
0.2500
0.3000
```