1300 - Taxes

It’s easy to think we can use binary-search to get the total income, and check whether it is accordant. But the main difficult is to avoid precious error! That depends on less of your skill but much of your luck! I’ll have some suggestion to you depending on my experience.

“The calculation of the regional coefficient and all kinds the taxes is made with rounding off to two digits after a decimal point. The initial income is given within two digits after a decimal point too.”

My way to achieve it is to hold two digits after decimal point, after my counting of function T(K).

Using long double (a bit like extended in pascal), I failed on case 7 again and again. Finally, I used long long (int64 in Pascal). I multiplied the value by 1000, and record it in long long type.

In the binary-search, my *t*/*w* is [0, 100000000000000]. This value is well-founded. There’s a fantastic line in the problem description :D

“All the parameters are integer not greater than 10^9 within two digits after a decimal point.”

I have nothing to say but faint. The correct understanding is that all the parameters are less than 10^9 within…

After my changing it to long long, the value doesn’t exceed $10^9*10^3=10^{12}$. Coz the net income is always larger than 1% of the total income (all the numbers in array S is 0~99), so 10^{12} / 1% = 10^{14}. Someone might say why can’t *w* be larger? We must avoid the limit exceeding in function T(K)!

After the binary-search, we’d better check the answer’s +1 or -1, and choose the best.

PS: the test case 8’s elements in S and L are all 0.

1301 - Cube in Labyrinth

What a pity. The testdata for it kept been wrong before 2 days after the contest. During the contest we failed on it again and again. The standard output for “No Solution” was changed to “no” during that time. I’ve tried -2~100 instead of “No Solution”, but all failed.. Unluckily, I forgot to check “no” :(.

The way to solve this problem is to use search, and use hash to judge the repeats of states. Use f[i,j,c,dire] to denote a state - (i,j) is the position, the upper face of dice is c (1..6) and dire denotes of 4 different rotate of it.

1302 - Delta-wave

Split one line into two (by odds and evens). Then we could count the “coordinate” of two numbers. Then everything is easy.

1303 - Minimal coverage

Sort all of the segments by its left end. Use O(N) in time to use greedy algorithm after it :D

1304 - Parallelepiped

Choose any of two points, set one to the maximum Z coordinate, the other to the minimum Z coordinate. After that, we can get the points between them, that is a easier problem in PLANE.

Finding the maximal rectangle in plane is a classical problem, with complexity $O(N^2)$ in time. How to do that? Enumerate the point on its left edge (or maybe its left edge is just the left edge of the plane), then scan from left to right. (of coz you must sort them before all work), update two variable MaxY and MinY. Clear readers must have known the algorithm well I think.

1305 - Convex hull

The convex polygon must be octagon, so we should use O(N) in time to do some statistic, such as the MaxX, MaxY…

But there’re many special cases should be pay more attention to.

1306 - Sequence Median

The may difficulty in this problem is to avoid the high space complexly. There’s a good way to make it in only O(N/2). Use a maximum heap, with exact N/2+1 elements. Haha~ Do you get it?

1307 - Archiver

It is very different between C/C++ and Pascal for this problem. I’m using C++.

My way to solve this problem is to zip 8bytes to 7, coz the visible characters in the Text is only 32~128 + 10, but the visible characters in programming is 32~254. Let analyze the binary digits for the last 1 byte, it is up to 7bits, if the ith bit is 1, then we can add 120 into the ith byte.

For C++ers, you must pay more attention to “\n” and “\\”! Then you must get Aced.

1308 - Deans pyramid

Algorithm and Pictures provided by Jin Kai in Hunan province, China. First you should download the 3 bmps here.

First let's see **bmp1**, the only conditions that can be reached is like the bmp, or rotate it with 90 degrees.

Let's see **bmp2**, we can know it well with this picture..

Finally we come to **bmp3**, the pink lines denotes the coordinate of them, besides each "integral point", there're 9 states that can be reached from the initial state, numbered from 1 to 9.

1309 - Dispute

Suppose f(1)=a1*f(0)+b1 (mod 9973), then f(9973*i+1)==a1*f(9973*i)+b1 (mod 9973).

So after calculating the (a,b) in f(9973)=a*f(0)+b (mod 9973), then f(9973*(i+1))=a*f(9973)+b (mod 9973)

So we can get the algorithm with O(10000) in time.

1310 - ACM Diagnostics

You might know that the way to solve this problem is like dp. Use f[i, j] to denote the number of ways, with i numbers, and its sum mod K = j.

f[i,j]=Sigma{f[i-1,(j-t) mod k],t=1,2,...,M}

This is an O(LKM) algorithm, coz we must use bignum, so it is very dangerous. We come up with a good idea:

f[i,j]=f[i,j-1]+f[i-1,(j-1) mod k]-f[i-1,(j-1-m) mod k]

So we can count f[i, 0] in O(M), and f[i,j] can be counted in O(1) each.

The complexity in time is O(L(M+K)) now.

1311 - Bricks

I misunderstood the problem at first. Notice each tile can be put on at most one tile.

Using greedy algorithm, from the bottom to top, count each tile's center of gravity, and check whether it is inside its underprop's edge. If it is, then unite them and count the new center of gravity, or else, the problem must exit with "impossible".

1312 - Tray

For a fixed table:

```
+----------+

|         |

|         |

|         |

|         |

+----------+
```

The largest plate must be put on the left-bottom corner. Notice that there're two ways to put the table, horizontal or vertical. The second largest plate must be put on the right edge, and as close to the bottom as possible. After that, we can use the given information of two plates, to denote the putable place of the last plate.

1313 - Some words about sport

I have nothing to say.

1314 - Chase in Subway

An interesting problem, first build the graph yourself!

For each of two points which must be passed through, suppose they are sour[0] and sour[1].

First let's get the shortest distance from sour[0] and sour[1] to every points else. For each point X, if it is connected with sour[0] and sour[1], and its distance to sour[1] + (m-1) is its distance to sour[0], then X might be the final points. Or else, i must NOT be the final points. Process all such pairs of points sour[0] and sour[1], then you'll AC..

1315 - MDPAR &amp; MIIAR

My algorithm is O(N*M*D) in time, but the testdatas are not strong enough:D. Just BFS as you like.

From the starting point, record the **current position** and **time that he didn't get air** in a queue. If the position was get to the second time, and the **time that he didn't get air** is smaller than before, then don't expand this node. 0.3sec AC.

1316 - Electronic Auction

You can use "interval tree" to be the data structure. There are many different structures that can be used.

1317 - Hail

For each hail, check the connecting segment between the hail and the lamp-source., if it is not enough high when crossing all of the fences.

1318 - Logarithm

Algorithm provided by Jin Kai in Hunan province, China.

$O(N^2)$ algorithm must get TLE coz the 128-bit numbers' operation. The main idea is to count the answer batch and batch.

Let's consider counting the number of pairs, whose result is more than X digit in decimal system. How to count so?

First write 10^X into binary system, suppose that is B. The new task is to count the number of pairs Ai xor Aj >= B.

Sort all of the Ai from small to large.

Consider each bit of B, the leftmost bit first. (*)

If it is 1, the corresponding bit of Ai and Aj must be 0 and 1.

If it is 0.

  If the corresponding bit of Ai and Aj is 0 and 1, the result must be larger than B.

  Or else (both 0 or both 1).

So it is obvious that we must build recursion system.

procedure getansone(dep,l,r:integer);

Compute the number of pairs that Ai xor Aj >= B, in bit=0~dep and for Ai, Aj, we have l<=i,j<=r.

procedure getanstwo(dep,l1,r1,l2,r2:integer);

Compute the number of pairs that Ai xor Aj >= B, in bit=0~dep and for Ai, Aj, we have l1<=i<=r1, l2<=j<=r2

getansone(127,1,N) is the one should be called in the main program. The idea of two procedures are the same, just as (*) describes:

First find the bend point of *dep*th bit, i.e. before this points, the *dep*th bits of all the numbers in interval [l,r] are 0, and after it, all of them are 1. (It seems that we can get the bend point using a good subalgorithm, but space… so just use binary-search as you like). After finding it, you can count and recur the procedure.

1319 - The Hotel

I have nothing to say.

1320 - Graph decomposition

You’re asked to judge whether the given graph (of coz the edges in it) can be split into some chains with two connected edges. Such as the sample input:

1 2

2 3

3 1

1 10

We can split it into two chains: 1-2-3 3-1-10

My algorithm is a little tricky, just check every connected component. If the number of edges in that component is even, then this component is “splitable”, or else, it is not. How to prove that? Mathematical induction is okay.

1321 - Floor indicator

The lowest digits can be got 0~9 easily, so we want to check the highest digit first: if the highest digit is

1: We must check 0~1

2~3: We must check 0~2

4~9: We must check 0~4

For the last two conditions, others digits will be checked wholly from 0 to 9, so the main difficulty remains on the first condition. For it, we should check the second highest digit. (I'll give you a hint, the two sub conditions depend on whether the second highest digit is larger than 1). Enjoy it.

1322 - Spy

The expanding version of IOI2001 binary code (spare problem). The algorithm is exactly simple, but you'll make mistake easily. Now I'll write the algorithm again:

For the example, we know that the last column is "rdarcaaaabb". We can simply sort them to "aaaaabbcdrr", this is the first column. Then we got some connection between them: r->a, d->a, a->a, r->a… But how about the order? "a" denotes the first "a" or the second "a"? In fact we can write them as:

a1 a2 a3 a4 a5 b1 b2 c1 d1 r1 r2

r1 d1 a1 r2 c1 a2 a3 a4 a5 b1 b2

Then we got it: a3 -> b2 -> r2 -> a4 -> c1 -> a5 -> d1 -> a2 -> b1 -> r1 -> a1.

How to prove it? It cannot be said clearly, but you should know that the order of the same letter in two sequences are strictly the same. Why? You can see it from this example:

abort < address < animal

and we also have:

borta < ddressa < nimala

1323 - Classmates

Enumerate all the possible order of N people, forming a sequence (of course the first one is monitor).

Then just like Bellman-ford, for every minute, from the last person, set the one he/she calls, to be the first person after him/her in the sequence (who hasn't received the message).

Find the best one among all of the possibilities.

1324 - Extra spaces

If we're asked to generate a solution for solving given L, without getting the largest K, we can simply use greedy. Each time, we process "x spaces to 1" operation. How large this x is? I make it near sqrt(L), and choose the one with smallest remaining spaces.

This is NOT the answer for this problem, it just gives the correct solution for solving given L, but we can run this program many times, and got some bend point, such as:

718052410 cost us 7 operations, 718052411 cost us 8 operations.

Then any one correct solution for 718052410 can be used as the answer for all of the input data, using "const" is fine!

Don't believe me? You'd better run your program to have a look :D

1325 - Dirt

An easy BFS problem, but you don't need to use priority queue. The easiest way is, for each time, generate from "i-times-changing-boots" states to "i+1-times-changing-boots" states. Then the complexity is low enough.

1326 - Bottle taps

It is easy to say the algorithm must have something to do with "hash dp", that means the status of dynamic programming should contain a 0~2^N-1 state, to denote the set of bought things. At first, I didn't find out the dynamic phase, but later I know:

X or Y >= X   in pascal,   (X|Y) >= X in c/c++

We can use f[i](i in 0~2^N-1) to denote, if we buy the things in set "i", the lowest price we should pay. Then from f[0] to f[2^N-1], expand them one by one.

1327 - Fuses (program experimentation by a275417 from Vietnam)

Discuss solely for each condition…

1328 - Fireball

WARNING: During the first N collisions, if the fireball has already touched point B, that touch is NOT considered as a "collision", so the fireball just fly through the ball smoothly.

I'm tested case 16 out, that's:

10 10

6

5 5 5 5

The correct answer is 45.00.

Now let’s enumerate the number of collisions in horizon and vertical. (The sum of them is N), and investigate the segment between them.

1329 - The galactic story

That’s a problem of LCA (least common ancestor), you could use the easiest LCA algorithm: O(n*log(n)) in both time and space. Just maintain an array p[i][j] to denote the **node i**’s ancestor in level 2^j.

The main difficult of this problem is to avoid MLE. So you should pay attention while you’re counting array p. (use as less unwanted memory as possible)

Another solution from a275417 (Vietnamese)

First label all the nodes with post-order traversal. You can then give the label interval of each sub-tree. After that, giving the label of any two nodes, you can check whether one’s label is inside the other’s interval. All the algorithm is O(N) in both time and space.

1330 - Intervals (program experimentation by a275417 from Vietnam)

Pre-calculate the sum of $A_1..A_i$.

1331 - Vladislava

My idea is not so good. And I could even give some data to make it TLE (but of coz I have other precaution to avoid it, but no need for URAL testdata. If you have better solution, please contact me)

I search the closing launching points for each artifact one by one. But I sort all of the points by latitude first. So I could cut a large number of branches down while searching.

Neal Zane’s another solution (Chinese):

closest <== least angle (0 <= angle <= 180) <== max. cosine value <== max. dot product (x1*x2+y1*y2+z1*z2)

so just convert all coordinates into 3-D x-y-z coordinates and for each artifact location, find the max. cosine value among the landing locations (M <= 5000). say the value is 'a', then the output should be: r * acos(a).

even enumeration can do it not too slowly because sine/cosine operations are eliminated.

1332 - Genie bomber

Enumerate two points on the circle, and calculate the center with the fixed radius. Check all of the circles calculated, and find the best one.

1333 - Genie bomber 2 (program experimentation by a275417 from Vietnam)

It doesn’t need a high precision calculation, so we can split the plane into small “pixels”, and do it one by one :).

1334 - Checkers

Be careful: the new checker might attack others and be attacked by others.

1335 - White Thesis (program experimentation by a275417 from Vietnam)

n*n+n   n*n+n+2   n*n+1

1336 - Problem of Ben Betsalel

(program experimentation by Thanh Vy from Vietnam)

$(n^2)^2 / n^3 = n$, with big number.

1337 - Bureaucracy

Be careful: there’s at most one bureaucrat on work each day! So simulate day by day.