CROATIAN OPEN COMPETITION IN INFORMATICS

7th ROUND

SOLUTIONS

Just carefully implement the "subtract 45 minutes" operation. The tricky parts are M < 45 and from 0:00 to 0:45.

Necessary skills:
simple number operations

Tags:
ad-hoc

First, note that the smallest possible size that contains K squares is the smallest power of 2 larger than or equal to K.
Greedy strategy yield the smallest number of breaks required to reduce 2^X to K. First we break the piece in two, yielding two pieces, both smaller than K, with 2^{x-1} squares each. We now take one of them and need K - 2^{x-1} squares more of the other one. We break that one in half and repeat the process. Each time we need more than half of the current square, we keep one part. On other occasion we throw that part away.

Necessary skills:
binary numbers, powers

Tags:
greedy algorithms

We precompute all distances between seats and rude passengers. We now repeatedly find the smallest global distance. We check to see what seat and passenger is in question. We find all other passengers equidistant to the seat, and than eliminate them all. We increment explosion counters as needed. Repeat until we run out of chairs, or passengers, or both.

Necessary skills:
shortest distance algorithm

Tags:
greedy algorithm

First, you need to know at least one way of constructing minimum spanning trees, for example Kruskal's algorithm:

- http://en.wikipedia.org/wiki/Kruskal%27s_algorithm

We claim that tunnels need to be constructed only between planets who are **immediate** neighbors on one of the axis. This can be easily proven. Suppose there are two planets (A and B) and that tunnel cost between A and B is $|x_A-x_B| \leq \min\{|y_A-y_B|, |z_A-z_B|\}$. Now suppose that A and B are not **immediate** neighbors on the X axis. In other words there is some planet C such that $x_A \leq x_C \leq x_B$. It is easy to see that instead of constructing one tunnel from A to B we can construct two tunnels: (A,C) and (C,B) that cost less than or equal to tunnel (A, b) since dist(A,C) + dist(C,B) $\leq |x_A-x_C| + |x_C-x_B| = |x_A-x_B|$.

We are nearly done now. We simply construct a graph of all planets with links between immediate neighbors. Using Kruskal's algorithm we obtain a minimal cost fully connected network.

Necessary skills:
Kruskal / Prim algorith, Minimum spanning tree

Tags:
graph theory

For start, note that simply calculating all pairs of distances is O(N^2) time complexity and scores 30% of points.
For 100% points, we use dynamic programming in O(R * S) time complexity. First, we find the sum of distances between all kings with coordinates $X_1 \le X_2$ and $Y_1 \le Y_2$, and on the second pass with $X_1 > X_2$ and $Y_1 < Y_2$. To find the sum we start at (0, 0) and work left to right / top to bottom. We need the following:
- `row_count(X, Y)`, `row_sum(X, Y)` - number of pieces in the fields with coordinates $X_i < X$, $Y = Y_i$ and the sum of distances to the field X, Y.
- `col_count(X, Y)`, `col_sum(X, Y)` - same as row but for columns
- `dp_count(X, Y)`, `dp_sum(X, Y)` - same but for the lower left part of the board, in other words for fields $X_i \le X$ and $Y_i \le Y$.

For dynamic programming we need the following relations:

```
row_sum(X, Y) = row_sum(X, Y-1) + row_count(X, Y-1)
row_count(X, Y) = row_count(X, Y-1) + B(X, Y)
col_sum(X, Y) = col_sum(X-1, Y) + col_count(X-1, Y)
col_count(X, Y) = col_count(X-1, Y) + B(X, Y)
dp_sum(X, Y) = dp_sum(X-1, Y-1) + dp_count(X-1, Y-1)
    + row_sum(X, Y) + col_sum(X, Y) + B(X, Y)
dp_count(X, Y) = dp_count(X-1, Y-1) + row_count(X, Y)
    + col_count(X, Y) + B(X, Y)
```

Where *B*(*X*, *Y*) is 1 if the field (*X*, *Y*) contains the figure of the current player, and 0 otherwise.
Now we can easily see that the sum of filed (X, Y) to other pieces is:

```
sum(X, Y) = dp_sum(X-1, Y-1) + row_sum(X, Y) + col_sum(X, Y),
```

Necessary skills:
dynamic programming

Tags:
dynamic programming

Degree of roads of any given city is the number of roads with endpoints in that city. First we create an additional city X, and connect it to all cities with odd degree of roads. It can be shown that this leaves no cities with odd degree (including X). This means that the graph is Eulerian, or if not connected, that each of its components is Eulerian. We now find Eulerian cycles in all components and allocate building rights on them alternating between 1 and 2.
For each component with at least one odd degreed city, this is a viable solution. For components with no odd degrees, there are two cases:

- One degree larger than or equal to 4. Starting from such degree forms a correct solution.
- All degrees equal to 2. The component is a cycle. If it contains an even number of cities, it is possible. Otherwise it is impossible to solve this graph.

Test cases:

Ther are 6 groups of test cases. First 4 groups are simple random graphs and are worth 20% of points. Next two groups are worth 40% each. The difference between two groups are graph sizes.
These are the descriptions of test cases in the larger group:

- a) large cycle, odd number of cities (impossible)
- b) tree
- c) chain
- d) large cycle, odd number of cities with a few extra cities connected to it
- e) 99 cycles with 1001 cities making a long chain and a couple of extra cities
- f) 10 cycles with 9999 cities making a long chain and one extra city
- g) random multi-component graph
- h) random multi-component graph with no degree larger than 3

Necessary skills:
Eulerov ciklus (there is another solution that doesn't require this)

Tags:
graph theory