NCPC 2011
Presentation of solutions

Heads of Jury: Jon Marius Venstad and Fredrik Svensson

2011-10-01

NCPC Jury

Problem authors

- Jon Marius Venstad (NTNU)
- Fredrik Svensson (Autoliv Electronics)
- Hans Wennborg (Google)
- Daniel Espling
- Lukáš Poláček (KTH)
- Serikzhan S. Kazi (Helsingin Yliopisto)
- Pål Grønås Drange (UiB)
- Roger Henriksson (Lund University)

C - Death Knight Hero

Solution

- Read through each line and check if the substring "CD" occurs anywhere on that line. If not, the hero scored a victory.

D – Elevator Trouble

Solution

This is a simple BFS exercise. Keep a queue (a linked list) of floors you can visit, remove `current`, and add (if not already visited) `current` $+ u$ and `current` $+ d$ to your queue, keeping track of how many steps you used to get there, i.e. `steps[current]`. Once you reach goal, you return `steps[goal]`. If you never reach the goal, you output `use the stairs`.
If you want, you can speed optimize by testing if the goal is above you and if $u = 0$, or if the goal is below you and $d = 0$, but this is strictly not necessary seeing the low number of steps needed.

F - Knigs of the Forest

Solution

- Karl-Älgtav will be able to win when a total of $k - 1$ weaker moose have entered the tournament.
- He also cannot win before he enters.
- Sort the moose that enter after 2011, and start counting the number of weaker moose each from 2011 and up.
- If you have gone through all the moose, and there still are less than $k - 1$ weaker moose, you cannot know when Karl-Älgtav will win.

G - Car Trouble

Solution

The street system forms a directed graph with cycles.

- Find, and output, streets trapping drivers by one of the following two methods:
    - Reverse the edges of the graph and traverse the graph from street 0 putting a mark on visited streets. Now, unmarked streets trap drivers.
    - For all $streets \neq 0$: Traverse the original graph starting from the street. If street 0 is not reached, the street traps drivers.
- Then, traverse the original graph starting at the street with $ID = 0$. Put a mark on visited streets. Output unmarked streets as unreachable.
- If no problems were found output "NO PROBLEMS".

The background story is said to describe a true event from Lund local politics.

A – Robots on a grid

Solution

Two problems in one. (1) counting the number of paths (beware of overflows), (2) a simple pathfinding exercise (using DFS or BFS). Counting is done when noticing that the number of paths from $(0,0)$ to (x,y) is given by

$$\mathtt{paths}(\mathtt{x},\mathtt{y}) = \mathtt{paths}(\mathtt{x}-1,\mathtt{y}) + \mathtt{paths}(\mathtt{x},\mathtt{y}-1)$$

The only pitfall is overflows. These usually occur when using

$$c = (a+b)\%\mathtt{val},$$

and $a+b$ overflow. Recall

$$(a+b)\%n = ((a\%n) + (b\%n))\%n$$

E - ls

Solution

- Write a simple recursive solution, similar to edit distance.
- Realise worst-case time complexity is very high.
- Use memoization or dynamic programming to reduce complexity to $O(|s||t|)$ for comparing strings s and t.

J - Enemy Division

Solution

- Two groups are always enough. Assign people arbitrarily into groups.
- While there is a soldier with two or more enemies in his group, change the group of that soldier.
- Why does this work? Consider the number of edges between enemies in a common group.
- Each time you do the above operation, the number of such edges is reduced by at least two, and increased by at most one. I.e. it decreases by at least one for each iteration.
- Since there are a maximum of $m = O(n)$ such edges in the first place, the algorithm has running time $O(n)$.

I - Prime Time

Solution

- Claiming a prime always allows the next player to claim 1.
- Primes are dense enough for the game not to go in loops.
- The results of claiming a number k may be found once all numbers reachable from k are finished.
- Find all primes up to 10007 using the Prime sieve of Erahosthenes.
- Fill out the results for all numbers from p_{i+1} down to $p_i + 1$, for $p_i = 2, 3, 5, 7, \ldots$.

H - Private Space

Solution

- First, add an imaginary person to each group to account for the free seats, and add an extra seat to each row.
- Try the following for each maximum row size $k = 2, 3, \ldots, 13$:
- Keep track of the number of rows r_i with i remaining seats.
- For each group of size $s = 13, 12, \ldots, 2$, try to seat that group at a row of i remaining seats, for each i.
- Use memoisation or dynamic programming to avoid repeatedly solving the same partial problems.

B - Mega Inversions

Solution

- Fenwick Trees keep track of the sum $\sum_{i=0}^{j} S_i$ for each j of an underlying array S_j, and can be updated in $\log(n)$ time.
- Let S_i be the given sequence, with $0 \leq i < n$.
- We want A to be a Fenwick Tree where position i gives the number of $S_j, j < i$ such that $S_j > S_i$.
- We want B to be a Fenwich Tree where position i gives the sum of entries $A_j, j < i$ such that $S_j > S_i$.
- Build A and B simultaneously by first adding 1 to A at position i_k, and then adding $A_{i_k - 1}$ to B at position i_k, for $k = 0, \ldots, n-1$.
- Do this in the order given by i_k such that $S_{i_k} \geq S_{i_{k-1}}$, and $i_k < i_{k-1}$ whenever $S_{i_k} = S_{i_{k-1}}$.
- The answer is the sum of $B_{i_k - 1}$ at each step k of the building procedure above.