ACM ICPC Shanghai Regional Solution

Just for fun

- Contest Organization: Donghua Univ.
- Problems and Tests: Lei Huang, Guodong Feng, Ji Hong
- Thanks to: Yonghui Wu, Zheng Sun, Kaiteng Huang, Xiaocheng Hu, Yi Yang, Mingyun Li

Problem Set

A: Alice’s Cube	479submits	115solved
B: Brute-force Algorithm	436submits	42solved
C: Compressed String	6submits	3solved
D: Decrypt Messages	11submits	4solved
E: Exciting Game	13submits	5solved
F: Flowers Placement	129submits	6solved
G: Game Simulator	0submits	0solved
H: Heroes Arrangment	7submits	0solved
I: Island Explorer	161submits	18solved
J: Jinyuetuan Puzzle	125submits	24solved

Alice’s Cube

- Easiest Problem
- Simple BFS or DFS
- The total number of states are only C(16,8)=12870
- We can use a 16-bit integer to memorize the states.
- Pre-calculation of all the reachable states will faster the algorithm

Brute-force Algorithm

- A math problem
- The problem is to calculate the power of Fibonacci number.
- Fermat-Euler theorem: When n is big enough

$$a^n \bmod P = a^{n \bmod \varphi(P) + \varphi(P)} \bmod P$$

- Then we can solve the problem by matrix multiplication.
- Or
- Find out the repeten and work out the answer directly and easily. (It’s easy to find in this problem)

Compressed String

- A tricky problem
- Brute-force to expand the string may not work well.
- We use hash to find the longest common prefix out.
- Binary search on the length of LCP, and check with hash.

Compressed String (Con’d)

- Calculate the hash value of the string:
- Value = s[0]+s[1]*a+s[2]*a^2+s[3]*a^3+…+s[n]*a^n % mod
- So that we can calc the hash value of the string repeat many times efficiently.

Decrypt Message

- This problem can be divided into two parts.
- The equation and the calendar
- Calendar is easy to implement.
- The equation part is quite tough.
- The standard solution try to find out one primitive root for the modulation.
- The algorithm to find out primitive root can be found in wikipedia.

Decrypt Message (Con’d)

- Suppose the primitive root is P. We can rewrite the equation as:
- P^(a*n)=P^(b)
- Then we can deal with a*n=b easily.

Exciting Time

- A time killer ☺
- Just simulate out the answer.
- First of all, one should memorize all the tetrominoes in the program.
- The standard solution use a lot of stacks and a STL-map to maintain the states.

Exciting Time(Con’d)

- When tetrominoes fall down. Just check the top of all the stacks and find out the height of the tetrominoes. Then push the blocks in the stacks.
- The map is to represent how many blocks in all the rows.
- If a row is full, we erase it in the STL-map.
- And only when the top of one stack is erased in the STL-map, we pop them respectively.

Flowers Placement

- The task is to find out the k-lexicographically binary match.
- We use DFS to work out the answer.
- Prune when there is no binary match since now.
- To speedup the prune, try not do Hungray algorithm on every node. We maintain the matching and find out the so-call argument circle to change from one solution to another.

Game Simulator

- A long description.
- A longer code.
- Just simulator the game of “tractor”
- A cryptic trick: When deal with “throw”, if one play three tractors with length 2, 2, and 3. And the next player play two tractors of length 3 and 4. If one use greedy algorithm to match the tractor of length 3 into the tractor of length 4. Then the remain tractor can’t be matching well. But the fact is opposite. 3=3, 4=2+2.
- We use some more discussion in the standard solution to avoid this problem.

Heroes Arrangement

- A 3D geometry task.
- To determine if one triangle obstruct another.
- We discuss some situation:
- 1. The points are obstruct points, edges or the triangles.
- 2. The edges are obstruct edges.
- Be careful with the graph building.
- Then an easy DFS or Bidirctional-search will work well.

Island Explorer

- An easy 2D geometry problem.
- For every point, just consider the previous and next point on the same line and the points near the vertical root on the other line.
- Then a simple kruskal algorithm will work.

Jinyuetuan Puzzle

- A naked dynamic programming.
- For every seconds, memorize the set of keys are pressing.
- Try to release some of the keys first and press some other keys.
- Be careful one can release keys first and then press keys, so that we can divide them into two second to simplify the problem.