ACM ICPC 2011 Asia Region Amritapuri Site

Onsite Editorials

A - Magic Grid

A - Magic Grid

Problem statement

- Given a RxC grid, Harry starts at (1,1) and the Sorcerer's stone is at (R,C)
- At each cell, Harry either gains/loses A[i][j] strength
- Find the minimum strength harry needs to start with, to collect the Sorcerer's Stone

Solution Idea (A - Magic Grid)

- If Harry starts with strength = S, can he reach (R,C) starting from (1,1) ?
- Can run a DP (Dynamic Programming) in row major order and check if S is enough
- Binary Search on the final answer S
- Can also be done using a single DP backwards

B - Save the Students

B - Save the Students

Problem statement

- Harry's spell can take the shapes of triangle, circle or square, and all who fall within that shape (including its boundaries) are protected.
- Find the number of people saved by Harry's spells.

Solution Idea
(B - Save the Students)

* Look at all the points in a suitable range and count points which lie within any of the shapes.
* Point in within a square with opposite corners (x1,y1) and (x2, y2), if x1 <= x <= x2, and y1 <= y <= y2.
* Point is within a circle, if it's distance from the center of the circle <= the radius of the circle.
* Point P is within a triangle ABC, the sum of the areas of triangles PAB,PAC,PBC should be equal to area of ABC.

* Tricky case: Some shape might be defined by positive integers, but it might encompass points with negative coordinates.

C - Robbing Gringotts

C - Robbing Gringotts

Problem statement

- Vault i contains X[i] gold items having weights of the gold items g[i][1],g[i][2],...,g[i][X[i]].
- Death Eater j has a bag which can hold weight v[j]. They can fill up his bag completely to its capacity by taking some subset of the objects present in a vault.
- Find the maximum weight of gold they can take away by planning their strategy correctly.

Solution Idea
(C - Robbing Gringotts)

* Two parts: First is to determine if Deatheater i will rob vault j. The second is determine the maximum gold they can get in an optimal assignment.
* For the first part (subset sum problem), use Meet-in-the-Middle.
* For the second part, use a Mincost Max Matching algorithm (hard to code).
* Alternatively for the second part, greedy bipartite matching possible after sorting the Deatheaters in descending order by their bag weights. (simple dfs based bipartite matching)
* Complexity: O(N * (M + |Xi|) * 2^(|Xi| / 2) + N * N * M).

D - Wizarding Duels

Solution Idea (D - Wizarding Duels)

* Sort the numbers.

* Sequence valid if for each i, (sum of all the array numbers from 0 to i) >= i * (i + 1) / 2. Also, total sum = n * (n - 1) / 2.

* DP with state (index, previous_number, current_sum) and O (1) transition - O(N^4) complexity.

* With state (index, previous_number, current_sum), you can greedily pick next number.
Low = max (previous_number, index * (index + 1) / 2)
High = (n * (n + 1) / 2 - current_sum) / (n - index)
At each step, take the closest number in [Low, High] and proceed greedily - O(N * logN) complexity.

* Can also be solved with max-flow.

E - Distinct Primes

Solution Idea (E - Distinct Primes)

* Iterate through all numbers from 1 and check which satisfy the condition (having at least 3 distinct prime factors) and output the n^{th}number

F - Magical Bridges

Problem statement & Solution Idea (F - Magical Bridges)

- Problem : Given a circular lane having N buildings and M bridges across their floors, answer a lot of shortest path

- Solution :
  - Imagine it as a graph with a node for each floor and edges between the floors directly connected
  - Observation : Only a very few nodes have degree > 2
  - Pick only those canonical nodes and run all pairs shortest path (Floyd-Warshalls O(N^3) fits in time)
  - Query : Shortest path between qfi and qfj
    - Each floor qfi can connect through canonical nodes only (at most two - one above and one below)
    - Binary Search for them

G - Here be Dragons

Solution Idea (G - Here be Dragons)

* Check if the input string has the character 'D' or not :)

H - Array Diversity

Problem statement & Solution Idea (H - Array Diversity)

Problem asks for lists containing the minimum and maximum
Part 1:
Counting number of subsequences which contain both the minimum and maximum. Tricky case when array contains only 1 distinct element - (4, 4, 4, 4)
The answer for general case is $(2^{count\_min}-1) * (2^{count\_max}-1) * 2^{rest}$

Answer for tricky case: 2^N-1

Take care with overflow and mod
The runtime of this algorithm is O(N) for counting and O(N) or O(logN) (using fast exponentiation) for computing the powers of 2.

Solution Idea (H - Array Diversity)

Part 2:
Counting number of substrings for array A[1..N]
Lets say that a particular substring starts at index i and ends at index j.
Now we iterate for i, and find the smallest j such that the segment A[i..j], A[i..j] must contain both the minimum and the maximum. Now the number of substrings starting at index i is f (i) = N-j+1. The final answer is sum of all f(i) for 1 <= i <= N
But this O(n^2) and we need something faster.
Now, lets say for index i, we know the index j. For index i+1, if the corresponding index is j', we can easily see that j <= j'
Thus, we can use a simple algorithm which maintains the count of min and max and updates j to j' when we increment i. The amortized runtime of this algorithm is be O(n).

I - Generations

Solution Idea (I - Generations)

* For each dragon c, compute d[i], which is the earliest birth year of a dragon born i generations later.
* We get k lists, where k is the number of children.
* Merge these k lists into one list cleverly by merging in O (smaller depth list) at each step.
* Binary search on the final merged list to get the answer for dragon c.
* Complexity: O(n log n).

* Alternate solution: Do a pre-order traversal so all descendants of any dragon are positioned contiguously.
* For each dragon, do range query to compute max depth amongst in that range amongst those which overlap with query interval.

J - Goblin Wars

Solution Idea (J - Goblin Wars)

* Do a BFS simultaneously with each of the civilizations as the starting point. The state is (x, y) which denote the coordinate of current cell.

* When transitioning between states, apart from the usual BFS conditions, note down the set of possible parents which can lead you to this current state for the same distance.

* Final observation is, for each state, you don't need to note down more than 2 parents. 0 parents - '.' ; 1 parent - character of parent ; >= 2 parents - '*'

* Complexity - O(R * C).

Trivia

* 1265 submissions
* 34 Queries asked
* 393 Balloons
* Easiest Problem : Problem G
* Toughest Problem : Problem D
* Max TLE : Problem J
* Max result: Wrong Answer 41.7%
* First (correct) Submission: Team Proof, Problem G (05:27)