CS3233 - Week 8

Problem A – Team Contest

Problem B – Spaceship

Mid-Contest:

Problem E – Pubs

Problem F – Garden

Problem G – Fire Station

Problem H – Sum

Problem I – Obstacle

Problem J – Bracket

Problem A – Team Contest

- Solution: DP
- Let a new team be formed (i, j, k).

  F[s] = max(F[s], 1 + F[s’]).

  s = the new state of used person.

  s’ = the old state of used person.

  s = s’ + (1 << i) + (1 << j) + (1 << k).

  s, s’ are bit-masks.

  For instance, if person0, person 1, person 2 are used, s = (1 << 0) + (1 << 1) + (1 << 2) = 7

Problem A – Team Contest

```
int dp(int used) {
    if (memo[used] != -1) return memo[used];
    int ret = 0;
    // first, find i, j, k, where all bits i, j, and k are off
    for (int i = 0; i < n; i++)
        if ((used & (1 << i)) == 0) {
            for (int j = i + 1; j < n; j++)
                if ((used & (1 << j)) == 0) {
                    for (int k = j + 1; k < n; k++)
                        if ((used & (1 << k)) == 0) {
                            // if these three programmers' abilities >= T,
                            // we can form a team with three of them
                            if (a[i] + a[j] + a[k] >= T) {
                                // try forming a team with them, turn on bits i, j, and k
                                ret = max(ret, 1 + dp(used | (1 << i) | (1 << j) | (1 << k)));
                            }
                        }
                }
        }
    return memo[used] = ret;
}
```

Problem B – Spaceship

- Solution: DP
- 2 states, id and remaining sphere (rs)
- if (do not use rs in trip-id)

$$F(id, rs) = F(id + 1, \min(rs + b[id], n)) + t[id];$$

if (use rs in trip-id, rs > 0)

$$F(id, rs) = \min(F(id, rs),$$
$$F(id + 1, \min(rs - 1 + b[id], n)) + t[id]/2)$$

Problem B – Spaceship

```
// id = current trip
// sphere = remaining sphere that you have (can be used for the next journey)
int dp(int id, int sphere) {
    if (id == n) return 0;
    if (memo[id][sphere] != -1) return memo[id][sphere];

    // do not use sphere
    memo[id][sphere] = t[id] + dp(id + 1, min(sphere + b[id], n));
    // we can use the sphere
    if (sphere) {
        // or use that sphere now
        memo[id][sphere] = min(memo[id][sphere],
                               (t[id] / 2) + dp(id + 1, min(sphere - 1 + b[id], n)));
    }
    return memo[id][sphere];
}
```

Problem E – Pubs

- Solution: Diameter of a tree
- Do 2 times BFS to determine the 2 vertices that should be picked.
- BFS huh? Why does it work? Uniform cost?

Problem E – Pubs

- It works because we are given a tree!
- Proof why 2 BFS works for finding a diameter of a tree:

http://apps.topcoder.com/forums/?module=Thread&threadID=668470&start=0&mc=12#1213839

- Algorithm: BFS from root, say node 0, you will get the deepest node, say node-i. From node-i, BFS 1 more time to get the deepest node, say node-j. The pubs lie in the path from node-i to node-j.
- After that, for each node, find the farthest distance to these pubs.

Problem F – Garden

- Solution: Offline Query + BIT (1 Dimensional)
- Offline Query: When you received a query, you do not have to answer it first. You can answer all queries after you have processed the queries.

Problem F – Garden

- Observe that you don't have to think about processing an arbitrary query "what is the sum in a (x1, y1, x2, y2) rectangle". Answering queries of the form "what is the sum of trees with x<=x1 and y<=y1" is all you need.

Problem F – Garden

- Sort both queries and trees by y-coordinate. Then process them one-by-one in order of increasing y. Each time you encounter a tree, add number x to your structure. Each time you encounter a query, count how many numbers <=x there are in your structure.

Problem G – Fire Station

- Solution: DP Tree
- Let best[x] = the minimum cost such that all the nodes in subtree-x can reach at least 1 fire station within the given distances.
- best[x] = min({f[x][i] | dist[i] <= d[x]}), where
  - d[x] = max distance of node-x to a fire station.
  - dist[i] = the distance of node-i to node-x. In other words, we set node-i as a new fire station.
  - f[x][i] = the cost of subtree at node-x with node-i as a new fire station.

Problem H – Sum

Sum of kth Powers of Divisors

$$\sigma_k(n) = (1 + p_1^k + p_1^{2k} + \cdots + p_1^{e_1 k})(1 + p_2^k + p_2^{2k} + \cdots + p_2^{e_2 k}) \cdots (1 + p_i^k + p_i^{2k} + \cdots + p_i^{e_i k})$$
$$= \prod_{a=1}^{i} \left(\sum_{b=0}^{e_a} p_a^{bk} \right)$$

where $p_1, p_2, \ldots, p_i$ are the distinct prime divisors of n.

Problem 1 – Obstacle

- Problem: Graph, check if the obstacles are connected from $y = 0$ to $y = W$ to block the car.
- First, we create a connectivity table between 2 nodes (2 obstacles). If the distance between the 2 obstacles (i and j) can block a car, we assign 1 to graph[i][j].
- We also consider the connectivity between the obstacles and the borders ($y = 0$ and $y = W$).

Problem I – Obstacle

- After that, we do dfs to check whether those obstacles can block the car.

Problem J – Bracket

- Solution: Stack
- We keep on the stack all the '(' that can still be matched with a close bracket. Then whenever we encounter a close bracket, we match it with the open bracket on top of the stack.
- While matching the '(' and ')', we can track the length of the regular bracket.