CS3233 - Week 9

Problem A – Finding A Wife

Problem B – Flood?

Problem C – Powers

Problem A – Finding A Wife

- Solution: Bipartite Matching or Max Flow
- Max Flow:
  - Connect **source** with the set of guys.
  - Connect the set of guys with the set of girls, connecting a node from guy set with a node from girl set only if this pair can be matched.
  - Connect the set of girls to **sink**.
  - Do max flow (Ford Fulkerson)

Problem A – Finding A Wife

- Code for Max Flow (refer to CP2)
- Code for Bipartite Matching:

```
bool dfs(int x) {
    for (int i = n; i < n + m; i++) {
        if (used[i]) continue;
        if (g[x][i]) {
            used[i] = 1;
            if (match[i] == -1 || dfs(match[i])) {
                match[i] = x;
                return 1;
            }
        }
    }
    return 0;
}
```

Problem A – Finding A Wife

```
while (scanf("%d%d", &n, &m) == 2) {
    if (n == 0 && m == 0) break;
    memset(g, 0, sizeof(g));
    for (int i = 0; i < n; i++) {
        scanf("%d", &t);
        for (int j = 0; j < t; j++) {
            scanf("%d", &x);
            g[i][x] = 1;
        }
    }
    memset(match, -1, sizeof(match));
    int ans = 0;
    for (int i = 0; i < n; i++) {
        memset(used, 0, sizeof(used));
        ans += dfs(i);
    }
    printf("%d\n", ans);
}
return 0;
```

Problem B – Flood?

- Solution: Max Flow (Ford Fulkerson, constraint is small), however you might need a fast max flow for final contest ;), e.g. Dinic (preferred), Pre-flow (optional)
- http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=maxFlow

Problem B – Flood?

- Connect **source** to locations of rain with flow = respective quantity.
- Connect location-i to location-j with flow = the value in Adjacency matrix [i][j]
- Connect locations connected to the sea with **sink**.

Problem C – Powers

- Skills Required: DP + Math
- There are many ways to solve this problem, but I will only explain one interesting solution.

Problem C – Powers

- Observe that:

$$(i+1)^k = \binom{k}{0} i^k + \binom{k}{1} i^{k-1} + \binom{k}{2} i^{k-2} + \cdots + \binom{k}{k-1} i^1 + \binom{k}{k} i^0$$

- Hence, we have the following relation:

$$\begin{bmatrix} \binom{k}{0} & \binom{k}{1} & \binom{k}{2} & \cdots & \binom{k}{k} \\ 0 & \binom{k-1}{0} & \binom{k-1}{1} & \cdots & \binom{k-1}{k-1} \\ 0 & 0 & \binom{k-2}{0} & \cdots & \binom{k-2}{k-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \binom{k-k}{0} \end{bmatrix} \cdot \begin{bmatrix} i^k \\ i^{k-1} \\ i^{k-2} \\ \vdots \\ i^0 \end{bmatrix} = \begin{bmatrix} (i+1)^k \\ (i+1)^{k-1} \\ (i+1)^{k-2} \\ \vdots \\ (i+1)^0 \end{bmatrix}$$

Problem C – Powers

- Next,

Let $a_i = 1^K + 2^K + \cdots + i^K$. The relation between a_i and a_{i+1} is

$$a_{i+1} = a_i + (i + 1)^K.$$

From this relation, we can build a matrix that allows us to compute a_{i+1} given a_i.

Problem C – Powers

$$\begin{bmatrix} 1 & \binom{k}{0} & \binom{k}{1} & \binom{k}{2} & \cdots & \binom{k}{k} \\ 0 & \binom{k}{0} & \binom{k}{1} & \binom{k}{2} & \cdots & \binom{k}{k} \\ 0 & 0 & \binom{k-1}{0} & \binom{k-1}{1} & \cdots & \binom{k-1}{k-1} \\ 0 & 0 & 0 & \binom{k-2}{0} & \cdots & \binom{k-2}{k-2} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \binom{k-k}{0} \end{bmatrix} \cdot \begin{bmatrix} a_i \\ i^k \\ i^{k-1} \\ i^{k-2} \\ \vdots \\ i^0 \end{bmatrix} = \begin{bmatrix} a_{i+1} \\ (i+1)^k \\ (i+1)^{k-1} \\ (i+1)^{k-2} \\ \vdots \\ (i+1)^0 \end{bmatrix}$$