

Contents

| | |
|---|-----------|
| 1 Sync | 1 |
| 1.1 Sync | 1 |
| 2 Data Structure | 1 |
| 2.1 Binary Search | 1 |
| 2.2 BIT | 1 |
| 2.3 BWT | 2 |
| 3 Divide and Conquer | 2 |
| 3.1 count inversions | 2 |
| 4 DP | 2 |
| 4.1 Doubling | 2 |
| 4.2 LCS | 2 |
| 4.3 LIS | 2 |
| 4.4 LIS 2 | 2 |
| 4.5 Minimum Edit Distance | 3 |
| 5 Enumerate | 3 |
| 5.1 Halfcut Enumerate | 3 |
| 6 Graph | 3 |
| 6.1 SPFA | 3 |
| 6.2 Dijkstra | 3 |
| 6.3 Floyd Warshall | 4 |
| 6.4 Disjoint set Kruskal | 4 |
| 6.5 Disjoint set Weight | 4 |
| 6.6 Bipartite 2 | 5 |
| 6.7 Hungarian algorithm | 5 |
| 6.8 LCA | 5 |
| 6.9 Trie | 6 |
| 7 Math | 6 |
| 7.1 Hash | 6 |
| 7.2 Math Theory | 6 |
| 7.3 Fibonacci | 6 |
| 7.4 Math | 7 |
| 7.5 Modular Multiplicative Inverse | 7 |
| 8 Function | 7 |
| 8.1 CHAR | 7 |
| 8.2 string | 7 |
| 8.3 setprecision | 7 |
| 8.4 GCD LCM | 7 |
| 8.5 reverse | 7 |
| 8.6 sort | 7 |
| 8.7 map | 8 |
| 8.8 set | 8 |
| 9 Other | 8 |
| 9.1 Ants Colony | 8 |
| 9.2 Binary codes | 9 |
| 9.3 Fire Fire Fire | 9 |
| 9.4 Disk Tree | 9 |
| 9.5 Stammering Aliens | 10 |
| 9.6 Fabled Rooks | 10 |
| 9.7 Rails | 10 |
| 9.8 String Distance and Transform Process | 11 |
| 10 Greedy | 11 |
| 10.1 Sticks | 11 |
| 11 DP | 12 |
| 11.1 Crested Ibis vs Monster | 12 |
| 11.2 dp Knapsack 1 | 12 |
| 11.3 Homer Simpson | 12 |
| 11.4 Let Me Count The Ways | 12 |
| 11.5 Luggage | 13 |
| 11.6 Partitioning by Palindromes | 13 |
| 11.7 SuperSale | 13 |
| 11.8 Walking on the Safe Side | 13 |
| 11.9 Cutting Sticks | 14 |
| 11.10 Race to 1 | 14 |
| 11.11 Apple | 14 |
| 11.12 Stamps | 15 |
| 11.13 Evacuation Plan | 15 |
| 11.14 Ladies Choice | 15 |
| 12 LIS | 16 |
| 12.1 Wavio Sequence | 16 |
| 12.2 Robots II | 16 |

| | |
|--------------------------------|-----------|
| 13 Math | 17 |
| 13.1 Big Mod | 17 |
| 13.2 Bubble Sort Expect Value | 17 |
| 13.3 Fraction Floor Sum | 17 |
| 13.4 How Many Os | 17 |
| 13.5 Number of Pairs | 18 |
| 13.6 ORXOR | 18 |
| 13.7 X drawing | 18 |
| 13.8 Playing With Stones | 18 |
| 13.9 And Then There Was One | 18 |
| 14 Binary Search | 18 |
| 14.1 Fill the Containers | 18 |
| 14.2 Where is the marble | 19 |
| 15 Graph | 19 |
| 15.1 Maximum sum on a torus | 19 |
| 16 Segment Tree | 19 |
| 16.1 Frequent values | 19 |
| 17 Dijkstra | 20 |
| 17.1 Airport Express | 20 |
| 17.2 Walk Through the Forest | 21 |
| 18 Kruskal | 21 |
| 18.1 Qin Shi Huang Road System | 21 |
| 19 Bipartite Graph | 22 |
| 19.1 Claw Decomposition | 22 |
| 19.2 Guardian of Decency | 22 |
| 19.3 Taxi Cab Scheme | 23 |
| 19.4 SAM I AM | 23 |

1 Sync

1.1 Sync

```
1 int main(){
2     std::ios::sync_with_stdio(false);
3     // 開始寫程式
4 }
```

2 Data Structure

2.1 Binary Search

```
1 int binary_search(int arr[maxn], int lef, int rig,
2     int target){
3     if(lef > rig) return 0x3f3f3f3f;
4     int mid = (lef + rig) >> 1;
5     if(arr[mid] == target) return mid;
6     else if(arr[mid] > target){
7         return binary_search(arr, lef, mid - 1,
8             target);
9     }
10    else{
11        return binary_search(arr, mid + 1, rig,
12            target);
13    }
14 }
```

2.2 BIT

```
1 /* BIT Binary Index Tree */
2 #define lowbit(k) (k & -k)
3 void add(vector<int> &tr, int id, int val) {
4     for (; id <= n; id += lowbit(id)) {
5         tr[id] += val;
6     }
7 }
8 int sum(vector<int> &tr, int id) {
9     int ret = 0;
10    for (; id >= 1; id -= lowbit(id)) {
11        ret += tr[id];
12    }
```

```

12 }
13 return ret;
14 }

```

2.3 BWT

```

1 /* BWT 資料轉換演算法 */
2 void BWT(){
3     for(int i = 0; i < n; ++i){
4         if(back[i] == 0)
5             mini[zero++] = i;
6     for(int i = 0; i < n; ++i)
7         if(back[i] == 1)
8             mini[zero++] = i;
9     int ptr = mini[0];
10    for(int i = 0; i < n; ++i){
11        cout << back[ptr] << " ";
12        ptr = mini[ptr];
13    }
14    cout << endl;
15 }

```

3 Divide and Conquer

3.1 count inversions

```

1 /*逆序數對*/
2 int arr[maxn], buf[maxn];
3 int count_inversions(int lef, int rig){
4     if(rig - lef <= 1) return 0;
5     int mid = (lef + rig)/2;
6     int ans = count_inversions(lef, mid) +
7               count_inversions(mid, rig);
8     int i = lef, j = mid, k = lef;
9     while(i < mid || j < rig){
10        if(i >= mid) buf[k] = arr[j++];
11        else if(j >= rig) buf[k] = arr[i++];
12        else{
13            if(arr[i] <= arr[j]) buf[k] = arr[i++];
14            else{
15                buf[k] = arr[j++];
16                ans += mid - i;
17            }
18            k++;
19        }
20    }
21    for(int k = lef; k < rig; ++k) arr[k] = buf[k];
22    return ans;
23 }

```

4 DP

4.1 Doubling

```

1 /* 倍增 */
2 int LOG = sqrt(N); // 2^LOG >= N
3 vector<int> arr(N);
4 vector<vector<int>> dp(N, vector<int>(LOG));
5 for(int i = 0; i < N; ++i) cin >> arr[i];
6 int L, Q, a, b;
7 cin >> L >> Q;
8 for(int i = 0; i < N; ++i){
9     dp[i][0] = lower_bound(arr.begin(), arr.end(),
10                             arr[i] + L) - arr.begin();
11     if(dp[i][0] == N || arr[i] + L < arr[dp[i][0]])
12         dp[i][0] = -1;
13 }
14 for(int i = 1; i < LOG; ++i)
15     for(int j = 0; j < N; ++j)

```

```

14     dp[j][i] = dp[dp[j][i - 1]][i - 1];
15 for(int i = 0; i < Q; ++i){
16     cin >> a >> b;
17     a--; // 要減減是因為arr的index從0開始但題目從1開始
18     b--;
19     if(a > b) swap(a, b);
20     int ans = 0;
21     for(int i = LOG - 1; i >= 0; --i){ // 從後往回推
22         if(dp[a][i] < b){
23             ans += (1 << i);
24             a = dp[a][i];
25         }
26     }
27     cout << ans + 1 << endl;
28 }

```

4.2 LCS

```

1 /* Longest Common Subsequence */
2 int LCS(string s1, string s2) {
3     int n1 = s1.size(), n2 = s2.size();
4     int dp[n1+1][n2+1] = {0};
5     // dp[i][j] = s1的前i個字元和s2的前j個字元
6     for (int i = 1; i <= n1; i++) {
7         for (int j = 1; j <= n2; j++) {
8             if (s1[i - 1] == s2[j - 1]) {
9                 dp[i][j] = dp[i - 1][j - 1] + 1;
10            } else {
11                dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
12            }
13        }
14    }
15    return dp[n1][n2];
16 }

```

4.3 LIS

```

1 /* Longest Increasing Subsequence */
2 int LIS(vector<int> &a) {
3     vector<int> s;
4     for (int i = 0; i < a.size(); i++) {
5         if (s.empty() || s.back() < a[i]) {
6             s.push_back(a[i]);
7         } else {
8             *lower_bound(s.begin(), s.end(), a[i],
9                           [](int x, int y) {return x < y;}) = a[i];
10        }
11    }
12    return s.size();
13 }

```

4.4 LIS 2

```

1 int LIS(vector<int> &a){
2     int len[a.size()];
3     for(int i = 0; i < a.size(); ++i) len[i] = 1;
4     int maxi = -1;
5     for(int i = 0; i < a.size(); ++i)
6         for(int j = i + 1; j < a.size(); ++j)
7             if(a[i] <= a[j]) len[j] = max(len[j],
8                                             len[i] + 1);
9     for(int i = 0; i < a.size(); ++i)
10        maxi = max(maxi, len[i]);
11    return maxi;
12 }

```

4.5 Minimum Edit Distance

```

1 // 利用 dfs 輸出替換字串的步驟
2 void backtracking(int i, int j){
3     if(i == 0 || j == 0){
4         while(i > 0){
5             cout << cnt++ << " Delete " << i << endl;
6             i--;
7         }
8         while(j > 0){
9             cout << cnt++ << " Insert " << i + 1 <<
10              " " << strB[j-1] << endl;
11             j--;
12         }
13         return;
14     }
15     if(strA[i-1] == strB[j-1]){
16         backtracking(i-1, j-1);
17     }
18     else{
19         if(dis[i][j] == dis[i-1][j-1] + 1){
20             cout << cnt++ << " Replace " << i << " " <<
21              strB[j-1] << endl;
22             backtracking(i-1, j-1);
23         }
24         else if(dis[i][j] == dis[i-1][j] + 1){
25             cout << cnt++ << " Delete " << i << endl;
26             backtracking(i-1, j);
27         }
28         else if(dis[i][j] == dis[i][j-1] + 1){
29             cout << cnt++ << " Insert " << i + 1 <<
30              " " << strB[j-1] << endl;
31             backtracking(i, j-1);
32         }
33     }
34 }
35 void MED(){
36     // 由於 B 是 0，所以 A 轉換成 B
37     // 時每個字元都要被刪除
38     for(int i = 0; i <= strA.size(); ++i) dis[i][0] =
39     i;
40     // 由於 A 是 0，所以 A 轉換成 B
41     // 時每個字元都需要插入
42     for(int j = 0; j <= strB.size(); ++j) dis[0][j] =
43     j;
44     for(int i = 1; i <= strA.size(); ++i){
45         for(int j = 1; j <= strB.size(); ++j){
46             // 字元相同代表不需修改，修改距離直接延續
47             if(strA[i-1] == strB[j-1]) dis[i][j] =
48             dis[i-1][j-1];
49             else{
50                 // 取 replace, delete, insert
51                 // 最小，選其 +1 為最少編輯距離
52                 dis[i][j] = min(dis[i-1][j-1],
53                 min(dis[i-1][j], dis[i][j-1])) +
54                 1;
55             }
56         }
57     }
58 }

```

5 Enumerate

5.1 Halfcut Enumerate

```

1 /* 折半枚舉 */
2 void dfs(set<long long int> &s, int depth, int T,
3         long long int sum){
4     if(depth >= T){
5         s.insert(sum);
6         return;
7     }
8     dfs(s, depth + 1, T, sum); // 取或不取的概念

```

```

8     dfs(s, depth + 1, T, sum + A[depth]);
9 }
10 int main(){
11     int N, T;
12     set<long long int> s1, s2;
13     cin >> N >> T;
14     for(int i = 0; i < N; ++i) cin >> A[i];
15     dfs(s1, 0, N/2, 0); // 折半枚舉
16     dfs(s2, N/2, N, 0);
17     long long int ans = 0;
18     // 題目:枚舉集合 Sx 的數字 Sxi, 找出 Sy
19     // 集合內小於等於 T-Sxi 中最大的數 Syj
20     for(auto &x : s1){
21         auto it = s2.upper_bound(T - x);
22         long long int y = *(--it);
23         if(x + y <= T) ans = max(ans, x + y);
24     }
25     cout << ans << endl;

```

6 Graph

6.1 SPFA

```

1 bool SPFA(int s){
2     // 記得初始化這些陣列
3     int cnt[1000+5], dis[1000+5];
4     bool inqueue[1000+5];
5     queue<int> q;
6
7     q.push(s);
8     dis[s] = 0;
9     inqueue[s] = true;
10    cnt[s] = 1;
11    while(!q.empty()){
12        int now = q.front();
13        q.pop();
14        inqueue[now] = false;
15
16        for(auto &e : G[now]){
17            if(dis[e.t] > dis[now] + e.w){
18                dis[e.t] = dis[now] + e.w;
19                if(!inqueue[e.t]){
20                    cnt[e.t]++;
21                    if(cnt[e.t] > m){
22                        return false;
23                    }
24                    inqueue[e.t] = true;
25                    q.push(e.t);
26                }
27            }
28        }
29    }
30    return true;
31 }

```

6.2 Dijkstra

```

1 /* Dijkstra 最短路徑 */
2 struct Edge{
3     int v, w;
4 };
5 struct Item{
6     int u, dis;
7     // 取路徑最短
8     bool operator < (const Item &other) const{
9         return dis > other.dis;
10    }
11 };
12 int dis[maxn];
13 vector<Edge> G[maxn];
14 void dijkstra(int s){

```

```

15   for(int i = 0; i <= m; i++){
16       dis[i] = inf;
17   }
18   dis[s] = 0;
19   priority_queue<Item> pq;
20   pq.push({s, 0});
21   while(!pq.empty()){
22       // 取路徑最短的點
23       Item now = pq.top();
24       pq.pop();
25       if(now.dis > dis[now.u]){
26           continue;
27       }
28       // 把與 now.u 相連的點都跑一遍
29       for(Edge e : G[now.u]){
30           if(dis[e.v] > now.dis + e.w){
31               dis[e.v] = now.dis + e.w;
32               pq.push({e.v, dis[e.v]});
33           }
34       }
35   }
36 }

```

6.3 Floyd Warshall

```

1 void floyd_warshall(){
2     for(int i = 0; i < n; i++){
3         for(int j = 0; j < n; j++){
4             G[i][j] = INF;
5         }
6         G[i][i] = 0;
7     }
8     for (int k = 0; k < n; k++){ // 嘗試每一個中繼點
9         for (int i = 0; i < n; i++){ // 計算每一個i點與每一個j點
10            for (int j = 0; j < n; j++){
11                G[i][j] = min(G[i][j], G[i][k] + G[k][j]);
12            }
13        }
14    }
15 }

```

6.4 Disjoint set Kruskal

```

1 struct Edge{
2     int u, v;
3     double w;
4     bool operator < (const Edge &rhs) const{
5         return w < rhs.w;
6     }
7 }edge[maxn * maxn];
8 vector<Edge> G[maxn]; // 紀錄有哪些邊在 MST 上
9 int parent[maxn];
10 // disjoint set
11 int find(int x){
12     return x == parent[x] ? x : parent[x] = find(parent[x]);
13 }
14 bool unite(int a, int b){
15     int x = find(a);
16     int y = find(b);
17     if(x == y) return false;
18     parent[x] = y;
19     return true;
20 }
21 double kruskal(){
22     m = 0; // m: 邊的數量
23     for(int i = 0; i < n; ++i)
24         for(int j = i + 1; j < n; ++j)
25             edge[m++] = (Edge){i, j, dist(i, j)};
26     sort(edge, edge + m);

```

```

27     for(int i = 0; i < n; ++i){
28         parent[i] = i;
29         G[i].clear();
30     }
31     double total = 0.0;
32     int edge_cnt = 0;
33     for(int i = 0; i < m; ++i){
34         int u = edge[i].u, v = edge[i].v;
35         double cnt = edge[i].w;
36         if(unite(u, v)){
37             G[u].push_back((Edge){u, v, cnt});
38             G[v].push_back((Edge){v, u, cnt});
39             total += cnt;
40             if(++edge_cnt == n-1) break;
41         }
42     }
43     return total;
44 }

```

6.5 Disjoint set Weight

```

1 /* 帶權並查集 + 數論 */
2 const int maxn = 20000+5;
3 int n, Q, parent[maxn], value[maxn];
4 int find(int x){
5     if(parent[x] != x){
6         int tmp = parent[x];
7         parent[x] = find(parent[x]);
8         value[x] ^= value[tmp];
9     }
10    return parent[x];
11 }
12 bool unionSet(int x, int y, int v){
13     int xParent = find(x);
14     int yParent = find(y);
15     if(xParent == yParent) return (value[x] ^ value[y]) == v;
16     if(xParent == n) swap(xParent, yParent);
17     parent[xParent] = yParent;
18     value[xParent] = value[x] ^ v ^ value[y];
19     return true;
20 }
21 int main(){
22     int p, q, v, k, x, Case = 1;
23     while(scanf("%d %d", &n, &Q) && n && Q){
24         printf("Case %d:\n", Case++);
25         for(int i = 0; i <= n; ++i) parent[i] = i;
26         memset(value, 0, sizeof(value));
27         char str[100+5];
28         bool flag = false;
29         int facts = 0;
30         for(int i = 0; i < Q; ++i){
31             scanf("%s", str);
32             if(str[0] == 'I'){
33                 gets(str);
34                 facts++;
35                 if(flag) continue;
36                 int cnt = sscanf(str, "%d %d %d", &p, &q, &v);
37                 if(cnt == 2){
38                     v = q;
39                     q = n;
40                 }
41                 if(!unionSet(p, q, v)){
42                     flag = true;
43                     printf("The first %d facts are conflicting.\n", facts++);
44                 }
45             }
46             else{
47                 scanf("%d", &k);
48                 int ans = 0;
49                 bool check = true;
50                 map<int, int> mp;
51                 for(int j = 0; j < k; ++j){
52                     scanf("%d", &x);

```

```

53         if(flag) continue;
54         int xParent = find(x);
55         ans ^= value[x];
56         mp[xParent]++;
57     }
58     if(flag) continue;
59     map<int, int>::iterator it;
60     for(it = mp.begin(); it != mp.end(); it++){
61         if(it->second % 2){
62             if(it->first != n){
63                 check = false;
64                 break;
65             }
66             else ans ^= value[it->first];
67         }
68     }
69     if(check) printf("%d\n", ans);
70     else printf("I don't know.\n");
71 }
72 }
73 printf("\n");
74 }
75 }

```

6.6 Bipartite 2

```

1  /* 二分图匹配 + 最小点覆盖 */
2  const int maxn = 1000+5;
3  int R, C, N;
4  bool arr[maxn][maxn], visitX[maxn], visitY[maxn];
5  int matchX[maxn], matchY[maxn];
6  int dfs(int x){
7      visitX[x] = true;
8      for(int y = 1; y <= C; ++y){
9          if(arr[x][y] && !visitY[y]){
10             visitY[y] = true;
11             if(matchY[y] == 0 || dfs(matchY[y])){
12                 matchX[x] = y;
13                 matchY[y] = x;
14                 return 1;
15             }
16         }
17     }
18     return 0;
19 }
20 int Match(){
21     int sum = 0;
22     memset(matchX, 0, sizeof(matchX));
23     memset(matchY, 0, sizeof(matchY));
24     for(int i = 1; i <= R; ++i){
25         memset(visitX, false, sizeof(visitX));
26         memset(visitY, false, sizeof(visitY));
27         sum += dfs(i);
28     }
29     return sum;
30 }
31 int main(){
32     while(cin >> R >> C >> N && R && C && N){
33         memset(arr, false, sizeof(arr));
34         memset(visitX, false, sizeof(visitX));
35         memset(visitY, false, sizeof(visitY));
36         int row, col;
37         for(int i = 0; i < N; ++i){
38             cin >> row >> col;
39             arr[row][col] = true;
40         }
41         int cnt = Match();
42         cout << cnt;
43         memset(visitX, 0, sizeof(visitX));
44         memset(visitY, 0, sizeof(visitY));
45         for(int i = 1; i <= R; ++i){
46             if(matchX[i] == 0) dfs(i);
47         }
48         for(int i = 1; i <= R; ++i)
49             if(!visitX[i]) cout << " r" << i;
50         for(int i = 1; i <= C; ++i)

```

```

50         if(visitY[i]) cout << " c" << i;
51         cout << endl;
52     }
53 }

```

6.7 Hungarian algorithm

```

1  /* 匈牙利演算法 */
2  const int maxn = 500+5;
3  int t, N, bn, gn, match[maxn];
4  bool visited[maxn];
5  vector<vector<int>> G(maxn);
6  struct People{
7      int h;
8      string music, sport;
9      People(){}
10     People(int h, string music, string sport){
11         this->h = h;
12         this->music = music;
13         this->sport = sport;
14     }
15 }lef[maxn], rig[maxn];
16 bool check(People boy, People girl){
17     if(abs(boy.h - girl.h) <= 40 && boy.music ==
18         girl.music && boy.sport != girl.sport) return
19         true;
20     return false;
21 }
22 bool dfs(int s){
23     for(int i = 0; i < G[s].size(); ++i){
24         int v = G[s][i];
25         if(visited[v]) continue;
26         visited[v] = true;
27         if(match[v] == -1 || dfs(match[v])){
28             match[v] = s;
29             return true;
30         }
31     }
32     return false;
33 }
34 int Hungarian(){
35     int cnt = 0;
36     memset(match, -1, sizeof(match));
37     for(int i = 0; i < bn; ++i){
38         memset(visited, false, sizeof(visited));
39         if(dfs(i)) cnt++;
40     }
41     return cnt;
42 }
43 int main(){
44     cin >> t;
45     while(t--){
46         cin >> N;
47         bn = 0, gn = 0;
48         for(int i = 0; i <= N; ++i) G[i].clear();
49         int h;
50         string sex, music, sport;
51         for(int i = 0; i < N; ++i){
52             cin >> h >> sex >> music >> sport;
53             if(sex == "M") lef[bn++] = People(h,
54                 music, sport);
55             else rig[gn++] = People(h, music, sport);
56         }
57         for(int i = 0; i < bn; ++i){
58             for(int j = 0; j < gn; ++j)
59                 if(check(lef[i], rig[j]))
60                     G[i].emplace_back(j);
61         }
62         cout << N - Hungarian() << endl;
63     }
64 }

```

6.8 LCA

```

1  /*最低共同祖先*/
2  // 此 node 下有幾顆 node
3  int dfs(int node, int dep){
4      depth[node] = dep + 1;
5      if(G[node].empty()){
6          siz[node] = 1;
7          return 1;
8      }
9      int total = 1;
10     for(auto i : G[node])
11         total += dfs(i.v, dep + 1);
12     siz[node] = total;
13     return siz[node];
14 }
15 // 找出每個節點的 2^i 倍祖先
16 // 2^20 = 1e6 > 200000
17 void find_parent(){
18     for(int i = 1; i < 20; i++){
19         for (int j = 0; j < N; j++){
20             parent[j][i] =
                parent[parent[j][i-1]][i-1];
21         }
22     }
23     // 求兩點的LCA (利用倍增法)
24     int LCA(int a, int b){
25         if (depth[b] < depth[a]) swap(a, b);
26         if (depth[a] != depth[b]){
27             int dif = depth[b] - depth[a];
28             for (int i = 0; i < 20; i++){
29                 if (dif & 1) b = parent[b][i];
30                 dif >>= 1;
31             }
32             if (a == b) return a;
33             for (int i = 19; i >= 0; i--){
34                 if (parent[a][i] != parent[b][i]){
35                     a = parent[a][i];
36                     b = parent[b][i];
37                 }
38             }
39             return parent[a][0];
40 }

```

6.9 Trie

```

1  /* Trie 字典樹 */
2  struct Tire{
3      int path;
4      map<string, int> G[maxn];
5      void init(){
6          path = 1;
7          G[0].clear();
8      }
9      void insert(string str){
10         int u = 0;
11         string word = "";
12         for(int i = 0; i < str.size(); ++i){
13             if(str[i] == '\\'){
14                 if(!G[u].count(word)){
15                     G[path].clear();
16                     G[u][word] = path++;
17                 }
18                 u = G[u][word];
19                 word = "";
20             }
21             else word += str[i];
22         }
23     }
24     void put(int u, int space){
25         for(auto i = G[u].begin(); i != G[u].end(); ++i){
26             for(int j = 0; j < space; ++j){
27                 cout << " ";
28             }
29             cout << i->first << endl;
30             put(i->second, space + 1);

```

```

31     }
32 }
33 }tree;

```

7 Math

7.1 Hash

```

1  /* 建議搭配 Other - Stammering Aliens 食用 */
2  #define ull unsigned long long int
3  const int maxn = 400005;
4  const ull seed = 131;
5  ull pw[maxn], hhash[maxn], hhash2[maxn];
6  char str[maxn];
7  void init(){
8      hhash[0] = 0;
9      for(int i = len-1; i >= 0; --i)
10         hhash[i] = (hhash[i+1] * seed + str[i]);
11 }

```

7.2 Math Theory

- Inversion 反轉:
 $aa^{-1} \equiv 1 \pmod{m}$. a^{-1} exists iff $\gcd(a, m) = 1$.
- Linear inversion 線性逆推:
 $a^{-1} \equiv (m - \lfloor \frac{m}{a} \rfloor) \times (m \bmod a)^{-1} \pmod{m}$
- Fermat's little theorem 费马小定理:
 $a^p \equiv a \pmod{p}$ if p is prime.
- Euler function 歐拉函數:
 $\phi(n) = n \prod_{p|n} \frac{p-1}{p}$
- Euler theorem 欧拉定理:
 $a^{\phi(n)} \equiv 1 \pmod{n}$ if $\gcd(a, n) = 1$.
- Extended Euclidean algorithm 擴展歐幾里得演算法:
 $ax + by = \gcd(a, b) = \gcd(b, a \bmod b) = \gcd(b, a - \lfloor \frac{a}{b} \rfloor b) = bx_1 + (a - \lfloor \frac{a}{b} \rfloor b)y_1 = ay_1 + b(x_1 - \lfloor \frac{a}{b} \rfloor y_1)$
- Divisor function 除数函数:
 $\sigma_x(n) = \sum_{d|n} d^x$. $n = \prod_{i=1}^r p_i^{a_i}$.
 $\sigma_x(n) = \prod_{i=1}^r \frac{p_i^{(a_i+1)x} - 1}{p_i^x - 1}$ if $x \neq 0$. $\sigma_0(n) = \prod_{i=1}^r (a_i + 1)$.
- Chinese remainder theorem 中國餘數定理:
 $x \equiv a_i \pmod{m_i}$.
 $M = \prod m_i$. $M_i = M/m_i$. $t_i = M_i^{-1}$.
 $x = kM + \sum a_i t_i M_i$, $k \in \mathbb{Z}$.

7.3 Fibonacci

$$f(n) = f(n-1) + f(n-2)$$

$$\begin{bmatrix} f(n) \\ f(n-1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{(n-1)} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$O(\log n)$$

```

1  LL fib(int n) {
2      if (n <= 1) {
3          return n;
4      }
5      Matrix a(2, 2), b(2, 1);
6      a.v[0][0] = a.v[0][1] = a.v[1][0] = 1;
7      b.v[0][0] = 1;
8      auto t = mPow(a, n - 1);
9      t = t * b;
10     return t.v[0][0];
11 }

```

7.4 Math

```

1 // 中国剩余定理
2 LL CRT(int k, LL* a, LL* r) {
3     LL n = 1, ans = 0;
4     for (int i = 1; i <= k; i++) n = n * r[i];
5     for (int i = 1; i <= k; i++) {
6         LL m = n / r[i], b, y;
7         exgcd(m, r[i], b, y); // b * m mod r[i] = 1
8         ans = (ans + a[i] * m * b % n) % n;
9     }
10    return (ans % n + n) % n;
11 }
12
13 // 模意义下取幂
14 long long binpow(long long a, long long b, long long
15 m) {
16     a %= m;
17     long long res = 1;
18     while (b > 0) {
19         if (b & 1) res = res * a % m;
20         a = a * a % m;
21         b >>= 1;
22     }
23     return res;
24 }

```

7.5 Modular Multiplicative Inverse

```

1 // 乘法逆元
2 // c++
3 void exgcd(int a, int b, int& x, int& y) {
4     if (b == 0) {
5         x = 1, y = 0;
6         return;
7     }
8     exgcd(b, a % b, y, x);
9     y -= a / b * x;
10 }
11
12 // python
13 def exgcd(a, b):
14     if b == 0:
15         x = 1
16         y = 0
17         return x, y
18     x1, y1 = exgcd(b, a % b)
19     x = y1
20     y = x1 - (a // b) * y1
21     return x, y

```

8 Function

8.1 CHAR

```

1 isdigit()
2 isalnum() // 判断字母 // 数字
3 isalpha()
4 islower()
5 isupper()
6 isblank() // 判断即 space 和 \t
7 toupper()
8 tolower()

```

8.2 string

```

1 int main(){
2     string str;
3     while(cin >> str){

```

```

4         // substr 取 str idx 2~4 的值
5         cout << str.substr(2, 4) << endl;
6         // substr 取 str idx 2 以後的所有值
7         cout << str.substr(2) << endl;
8
9         string subst;
10        cin >> subst;
11        // str.append 連接字串
12        cout << str.append(subst) << endl;
13
14        char s[100], ss[100];
15        cin >> s >> ss;
16
17        char *p;
18        // strstr 回傳在s裡找到ss後的整個字串(從 ss
19        // idx 0 到結束)
20        p = strstr(s, ss);
21        cout << p << endl;
22        // strstr 也可以單純用來找字串
23        if(p != NULL) cout << "yes" << endl;
24        else cout << "no" << endl;
25    }

```

8.3 setprecision

```

1 double cnt = 3.5555;
2 cout << fixed << setprecision(3) << cnt ;

```

8.4 GCD LCM

```

1 int gcd(int a, int b){
2     return (b == 0 ? a : gcd(b, a % b));
3 }
4 int lcm(int a, int b){
5     return a * b / gcd(a, b);
6 }
7
8 /* 輾轉相除法 - 求兩數是否互質
9 如果兩數互質 最終結果其中一方為0時 另一方必為1
10 若兩數有公因數 最終結果其中一方為0時 另一方必不為1 */
11 while ( ( num1 % num2 ) != 0 && ( num2 % num1 ) != 0 );

```

8.5 reverse

```

1 int a[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
2 reverse(a, a + 5);
3
4 vector<int> v;
5 reverse(v.begin(), v.end());
6
7 string str = "123";
8 reverse(str.begin(), str.end());
9 cout << str << endl; //321

```

8.6 sort

```

1 priority_queue<int, vector<int>, less<int>> // 大到小
2 priority_queue<int, vector<int>, greater<int>> //
3     小到大
4 int arr[] = {4, 5, 8, 3, 7, 1, 2, 6, 10, 9};
5 sort(arr, arr+10);
6
7 vector<int> v;
8 sort(v.begin(), v.end()); //小到大
9

```

```

10 int cmp(int a, int b){
11     return a > b;
12 }
13 sort(v.begin(), v.end(), cmp); //大到小

```

8.7 map

```

1 int main(){
2     map<string, string> mp;
3     map<string, string>::iterator iter;
4     map<string, string>::reverse_iterator iter_r;
5
6     mp.insert(pair<string, string>("r000", "zero"));
7
8     mp["r123"] = "first";
9
10    for(iter = mp.begin(); iter != mp.end(); iter++)
11        cout<<iter->first<<" "<<iter->second<<endl;
12    for(iter_r = mp.rbegin(); iter_r != mp.rend();
13        iter_r++)
14        cout<<iter_r->first<<" "
15            <<iter_r->second<<endl;
16
17    iter = mp.find("r123");
18    mp.erase(iter);
19
20    iter = mp.find("r123");
21    if(iter != mp.end())
22        cout<<"Find, the value is "
23            <<iter->second<<endl;
24    else
25        cout<<"Do not Find"<<endl;
26
27    mp.clear();
28    mp.erase(mp.begin(), mp.end());
29 }

```

8.8 set

```

1 int main(){
2     set<int> st {1, 6, 8}; // 直接初始化的寫法
3     st.insert(1); // 也可以這樣寫就好
4     set<int>::iterator iter;
5
6     // 如果有找到，就會傳回正確的 iterator，否則傳回
7     // st.end()
8     if (iter != st.end()) {
9         cout << "Found: " << *iter << endl;
10    } else {
11        cout << "Not found." << endl;
12    }
13    // cout: Found: 6
14
15    // 取值：使用 iterator
16    x = *st.begin(); // set 中的第一個元素(最小的元素)
17    x = *st.rbegin(); // set
18    // 中的最後一個元素(最大的元素)
19
20    // search
21    iter = st.find(6);
22    auto it = st.find(x); // binary search, O(log(N))
23    auto it = st.lower_bound(x); // binary search,
24    // O(log(N))
25    auto it = st.upper_bound(x); // binary search,
26    // O(log(N))
27
28    st.clear();
29 }

```

9 Other

9.1 Ants Colony

```

1 /* LCA 最低共同祖先 */
2 const int maxn = 1e5 + 5;
3 struct Edge{
4     int v;
5     int w;
6 };
7 int N;
8 vector<Edge> G[maxn];
9 int parent[maxn][20+5];
10 int depth[maxn], siz[maxn];
11 // 此 node 下有幾顆 node
12 int dfs(int node, int dep){
13     depth[node] = dep + 1;
14     if(G[node].empty()){
15         siz[node] = 1;
16         return 1;
17     }
18     int total = 1;
19     for(auto i : G[node])
20         total += dfs(i.v, dep + 1);
21     siz[node] = total;
22     return siz[node];
23 }
24 // 找出每個節點的 2^i 倍祖先
25 // 2^20 = 1e6 > 200000
26 void find_parent(){
27     for(int i = 1; i < 20; i++){
28         for (int j = 0; j < N; j++){
29             parent[j][i] =
30                 parent[parent[j][i-1]][i-1];
31         }
32     }
33 }
34 // 求兩點的 LCA (利用倍增法)
35 int LCA(int a, int b){
36     if (depth[b] < depth[a]) swap(a, b);
37     if (depth[a] != depth[b]){
38         int dif = depth[b] - depth[a];
39         for (int i = 0; i < 20; i++){
40             if (dif & 1) b = parent[b][i];
41             dif >>= 1;
42         }
43     }
44     if (a == b) return a;
45     for (int i = 19; i >= 0; i--){
46         if (parent[a][i] != parent[b][i]){
47             a = parent[a][i];
48             b = parent[b][i];
49         }
50     }
51     return parent[a][0];
52 }
53 long long int dist[maxn];
54 // 從 0 開始到每個點的距離
55 void distance(){
56     for (int u = 0; u < N; ++u){
57         for(int i = 0; i < G[u].size(); ++i){
58             dist[G[u][i].v] = dist[u] + G[u][i].w;
59         }
60     }
61 }
62 int main(){
63     while(cin >> N && N){
64         memset(dist, 0, sizeof(dist));
65         memset(parent, 0, sizeof(parent));
66         memset(depth, 0, sizeof(depth));
67         memset(siz, 0, sizeof(siz));
68         for(int i = 0; i <= N; ++i){
69             G[i].clear();
70         }
71         for(int i = 1; i < N; ++i){
72             int u, w;
73             cin >> u >> w;
74             G[u].push_back({i, w});
75             parent[i][0] = u;
76         }
77     }
78 }

```



```

71     }
72     find_parent();
73     dfs(0, 0);
74     distance();
75     int s; cin >> s;
76     bool space = false;
77     for(int i = 0; i < s; ++i){
78         int a, b;
79         cin >> a >> b;
80         int lca = LCA(a, b);
81         if(space) cout << " ";
82         space = true;
83         cout << (dist[a] + dist[b]) - (dist[lca]
84             * 2);
85     }
86     cout << endl;
87 }

```

9.2 Binary codes

```

1  /* BWT 資料轉換演算法 */
2  void BWT(){
3      for(int i = 0; i < n; ++i){
4          if(back[i] == 0){
5              mini[zero++] = i;
6          }
7          for(int i = 0; i < n; ++i){
8              if(back[i] == 1){
9                  mini[zero++] = i;
10             }
11             int ptr = mini[0];
12             for(int i = 0; i < n; ++i){
13                 cout << back[ptr] << " ";
14                 ptr = mini[ptr];
15             }
16             cout << endl;
17         }
18     }
19     int main(){
20         cin >> n;
21         for(int i = 0; i < n; ++i){
22             cin >> back[i];
23             zero = 0;
24             BWT();
25         }
26     }

```

9.3 Fire Fire Fire

```

1  /* dfs
2  只要我有一個小孩不是防火牆，我就必須是防火牆 */
3  #include <bits/stdc++.h>
4  using namespace std;
5  const int maxn = 1000+5;
6  int cnt = 0;
7  vector<int> G[maxn];
8  bool exi[maxn], visited[maxn];
9  void dfs(int node, int parent){
10     if(G[node].size() == 1 && G[node][0] == parent)
11         return;
12     for(int i = 0; i < G[node].size(); ++i){
13         int now = G[node][i];
14         if(visited[now]) continue;
15         visited[now] = true;
16         dfs(G[node][i], node);
17     }
18     bool flag = false;
19     for(int j = 0; j < G[node].size(); ++j){
20         if(exi[G[node][j]] != true && G[node][j] !=
21             parent){
22             flag = true;
23             break;
24         }
25     }
26     if(flag && exi[node] != true){
27         exi[node] = true;
28     }
29 }

```

```

26     cnt++;
27 }
28 return;
29 }
30 int main(){
31     int n;
32     while(cin >> n && n){
33         for(int i = 1; i <= n; ++i) G[i].clear();
34         memset(exi, false, sizeof(exi));
35         memset(visited, false, sizeof(visited));
36         for(int i = 1; i <= n; ++i){
37             int siz;
38             cin >> siz;
39             for(int j = 0; j < siz; ++j){
40                 int num;
41                 cin >> num;
42                 G[i].emplace_back(num);
43             }
44         }
45         cnt = 0;
46         dfs(1, 1);
47         if(n == 1) cnt++;
48         cout << cnt << endl;
49     }
50 }

```

9.4 Disk Tree

```

1  /* Trie 字典樹 */
2  const int maxn = 50000+5;
3  struct Tire{
4      int path;
5      map<string, int> G[maxn];
6      void init(){
7          path = 1;
8          G[0].clear();
9      }
10     void insert(string str){
11         int u = 0;
12         string word = "";
13         for(int i = 0; i < str.size(); ++i){
14             if(str[i] == '\\'){
15                 if(!G[u].count(word)){
16                     G[u].clear();
17                     G[u][word] = path++;
18                 }
19                 u = G[u][word];
20                 word = "";
21             }
22             else word += str[i];
23         }
24     }
25     void put(int u, int space){
26         for(auto i = G[u].begin(); i != G[u].end();
27             ++i){
28             for(int j = 0; j < space; ++j)
29                 cout << " ";
30             cout << i->first << endl;
31             put(i->second, space + 1);
32         }
33     }
34 }tree;
35 int main(){
36     int n;
37     string str;
38     while(cin >> n && n){
39         tree.init();
40         for(int i = 0; i < n; ++i){
41             cin >> str;
42             str += '\\';
43             tree.insert(str);
44         }
45         tree.put(0, 0);
46         cout << endl;
47     }

```

9.5 Stammering Aliens

```

1  /* hash 字串 + 二分搜尋 */
2  #define ull unsigned long long int
3  const int maxn = 40000+5;
4  const ull seed = 131;
5  ull pw[maxn], hhash[maxn], hhash2[maxn];
6  int m, len;
7  char str[maxn];
8  map<ull, int> mp;
9  void init(){
10     hhash[0] = 0;
11     for(int i = len-1; i >= 0; --i){
12         hhash[i] = (hhash[i+1] * seed + str[i]);
13     }
14 }
15 int check(int x){
16     for(int i = 0; i + x - 1 < len; ++i){
17         ull tmp = hhash[i] - (hhash[i + x] * pw[x]);
18         hhash2[i] = tmp;
19     }
20     sort(hhash2, hhash2 + len - x + 1);
21     int cnt = 0;
22     for(int i = 0; i < len - x + 1; ++i){
23         if(i && hhash2[i] == hhash2[i-1])
24             cnt++;
25         else{
26             if(cnt >= m) return 1;
27             cnt = 1;
28         }
29     }
30     if(cnt >= m) return 1;
31     return 0;
32 }
33 int main(){
34     pw[0] = 1;
35     for(int i = 1; i < maxn; ++i)
36         pw[i] = (pw[i-1] * seed);
37     while(scanf("%d", &m) && m){
38         scanf("%s", str);
39         len = strlen(str);
40         init();
41         int lef = 1, rig = len + 1;
42         while(lef < rig){
43             int mid = (lef + rig) >> 1;
44             if(check(mid))
45                 lef = mid + 1;
46             else rig = mid;
47         }
48         int ans = rig - 1;
49         if(!ans){
50             puts("none");
51             continue;
52         }
53         int pos;
54         mp.clear();
55         for(int i = 0; i + ans - 1 < len; ++i){
56             ull tmp = hhash[i] - hhash[i + ans] *
57                 pw[ans];
58             mp[tmp]++;
59             if(mp[tmp] >= m) pos = i;
60         }
61         printf("%d %d\n", ans, pos);
62     }
63     return 0;

```

```

7     int lef, rig, pos, idx;
8     bool operator < (const Edge &rhs) const{
9         if(rig != rhs.rig)
10             return rig < rhs.rig;
11         else
12             return lef < rhs.lef;
13     }
14 }x[maxn], y[maxn];
15 bool used[maxn];
16 bool solve_x(){
17     memset(used, false, sizeof(used));
18     for(int i = 0; i < n; ++i){
19         x[i].pos = 0;
20         for(int j = x[i].lef; j <= x[i].rig; ++j){
21             if(!used[j]){
22                 x[i].pos = j;
23                 used[j] = true;
24                 break;
25             }
26         }
27         if(x[i].pos == 0) return false;
28     }
29     return true;
30 }
31 bool solve_y(){
32     memset(used, false, sizeof(used));
33     for(int i = 0; i < n; ++i){
34         y[i].pos = 0;
35         for(int j = y[i].lef; j <= y[i].rig; ++j){
36             if(!used[j]){
37                 y[i].pos = j;
38                 used[j] = true;
39                 break;
40             }
41         }
42         if(y[i].pos == 0) return false;
43     }
44     return true;
45 }
46 int main(){
47     while(cin >> n && n){
48         int x1, y1, x2, y2;
49         for(int i = 0; i < n; ++i){
50             cin >> x1 >> y1 >> x2 >> y2;
51             x[i].lef = min(x1, x2);
52             x[i].rig = max(x1, x2);
53             y[i].lef = min(y1, y2);
54             y[i].rig = max(y2, y2);
55             x[i].idx = y[i].idx = i;
56             x[i].pos = y[i].pos = 0;
57         }
58         sort(x, x + n);
59         sort(y, y + n);
60         if(!solve_x() || !solve_y()) cout <<
61             "IMPOSSIBLE" << endl;
62         else{
63             int ans_x[maxn], ans_y[maxn];
64             for(int i = 0; i < n; ++i){
65                 ans_x[x[i].idx] = x[i].pos;
66                 ans_y[y[i].idx] = y[i].pos;
67             }
68             for(int i = 0; i < n; ++i)
69                 cout << ans_x[i] << " " << ans_y[i]
70                     << endl;
71         }
72     }
73 }

```

9.6 Fabled Rooks

```

1  /* 特定排序後放入格子
2  以右邊大小排序 要從左邊開始放
3  以左邊大小排序 要從右邊開始放 */
4  int n;
5  const int maxn = 5000+5;
6  struct Edge{

```

9.7 Rails

```

1  /* deque 火車
2  倒退逆推法 能怎樣進來就能怎樣出去
3  lis: 1 2 3 4 5
4  dq: 3 2 4 1 5
5  1. 如果 lis front = dq front, dq pop

```

```

6 2. 反之 ans.push dq front，每次檢查 ans top 是否 =
   lis front
7 */
8 int main(){
9     int n;
10    while(cin >> n && n){
11        int train;
12        deque<int> dq;
13        while(cin >> train && train){
14            dq.emplace_back(train);
15            deque<int> lis, ans;
16            for(int i = 2; i <= n; ++i){
17                cin >> train;
18                dq.emplace_back(train);
19            }
20            for(int i = 1; i <= n; ++i)
21                lis.emplace_back(i);
22            for(int i = 0, j = 0; j < n, i < n; ++i){
23                if(lis[i] == dq[j]) ++j;
24                else
25                    ans.emplace_back(lis[i]);
26                while(!ans.empty()){
27                    if(dq[j] != ans.back()) break;
28                    ans.pop_back();
29                    ++j;
30                }
31            }
32            if(!ans.empty())
33                cout << "No" << endl;
34            else
35                cout << "Yes" << endl;
36            dq.clear();
37        }
38        cout << endl;
39    }
40 }

```

9.8 String Distance and Transform Process

```

1 /* MED - Minimum Edit Distance
2 增加刪除修改 使得字串A 以最小步驟數替換成 字串B
3 abcac
4 bcd
5   j 0 b c d
6 i +-----
7 0 | 0 1 2 3
8 a | 1 1 2 3
9 b | 2 1 2 3
10 c | 3 2 1 2
11 a | 4 3 2 2
12 c | 5 4 3 3
13 1 Delete 1
14 2 Replace 3,d
15 3 Delete 4 */
16 const int maxn = 80+5;
17 string strA, strB;
18 int dis[maxn][maxn];
19 int cnt;
20 // 利用 dfs 輸出替換字串的步驟
21 void backtracking(int i, int j){
22     if(i == 0 || j == 0){
23         while(i > 0){
24             cout << cnt++ << " Delete " << i << endl;
25             i--;
26         }
27         while(j > 0){
28             cout << cnt++ << " Insert " << i + 1 <<
29                 ", " << strB[j-1] << endl;
30             j--;
31         }
32         return;
33     }
34     if(strA[i-1] == strB[j-1]){
35         backtracking(i-1, j-1);

```

```

36     else{
37         if(dis[i][j] == dis[i-1][j-1] + 1){
38             cout << cnt++ << " Replace " << i << ", "
39                 << strB[j-1] << endl;
40             backtracking(i-1, j-1);
41         }
42         else if(dis[i][j] == dis[i-1][j] + 1){
43             cout << cnt++ << " Delete " << i << endl;
44             backtracking(i-1, j);
45         }
46         else if(dis[i][j] == dis[i][j-1] + 1){
47             cout << cnt++ << " Insert " << i + 1 <<
48                 ", " << strB[j-1] << endl;
49             backtracking(i, j-1);
50         }
51     }
52 void MED(){
53     // 由於 B 是 0，所以 A 轉換成 B
54     // 時每個字元都要被刪除
55     for(int i = 0; i <= strA.size(); ++i) dis[i][0] = i;
56     // 由於 A 是 0，所以 A 轉換成 B
57     // 時每個字元都需要插入
58     for(int j = 0; j <= strB.size(); ++j) dis[0][j] = j;
59     for(int i = 1; i <= strA.size(); ++i){
60         for(int j = 1; j <= strB.size(); ++j){
61             // 字元相同代表不需修改，修改距離直接延續
62             if(strA[i-1] == strB[j-1]) dis[i][j] = dis[i-1][j-1];
63             else{
64                 // 取 replace, delete, insert
65                 // 最小，選其 +1 為最少編輯距離
66                 dis[i][j] = min(dis[i-1][j-1],
67                     min(dis[i-1][j], dis[i][j-1])) + 1;
68             }
69         }
70     }
71 }
72 int main(){
73     bool space = false;
74     while(getline(cin, strA) && getline(cin, strB)){
75         cnt = 1;
76         MED();
77         if(space) cout << endl;
78         space = true;
79         cout << dis[strA.size()][strB.size()] << endl;
80         backtracking(strA.size(), strB.size());
81     }
82 }

```

10 Greedy

10.1 Sticks

```

1 /* Greedy + dfs */
2 const int maxn = 100+5;
3 int n, stickLengthSum, ans, stick[maxn];
4 bool visited[maxn];
5 bool dfs(int length, int idx, int stickTotal){
6     if(length == ans){
7         if(stickTotal == n) return true;
8         length = 0;
9     }
10    if(length == 0){
11        for(idx = 0; visited[idx]; idx++);
12        visited[idx] = true;
13        if(dfs(length + stick[idx], idx+1,
14            stickTotal+1)) return true;
15        visited[idx] = false;
16    }
17    else{

```

```

17     for(int j = idx; j < n; ++j){
18         if(visited[j] || (j && stick[j] ==
19             stick[j-1] && !visited[j-1]))
20             continue;
21         if(stick[j] + length > ans) continue;
22         visited[j] = true;
23         if(dfs(length + stick[j], j+1,
24             stickTotal+1)) return true;
25         visited[j] = false;
26         if(length + stick[j] == ans) return false;
27     }
28     return false;
29 }
30 int main(){
31     while(scanf("%d", &n) && n){
32         stickLengthSum = 0;
33         for(int i = 0; i < n; ++i){
34             scanf("%d", &stick[i]);
35             stickLengthSum += stick[i];
36         }
37         sort(stick, stick + n, greater<int>());
38         for(ans = stick[0]; ans <= stickLengthSum;
39             ans++){
40             memset(visited, false, sizeof(visited));
41             if(stickLengthSum % ans != 0) continue;
42             if(dfs(0, 0, 0)) break;
43         }
44     }
45     printf("%d\n", ans);
46 }

```

11 DP

11.1 Crested Ibis vs Monster

```

1  /* dp 背包 - 重量/價值/可重複使用
2  9 3
3  8 3
4  4 2
5  2 1
6  0 3 3 3 3 3 3 3 6
7  0 2 2 2 2 3 3 3 5
8  0 1 1 2 2 3 3 3 4
9  因為這題可以重複使用同一條魔法
10 所以可以這樣 dp */
11 int a[10000+5], b[10000+5];
12 int dp[10000+5][10000+5];
13 int main(){
14     int h, n;
15     cin >> h >> n;
16     for(int i = 1; i <= n; i++){
17         cin >> a[i] >> b[i];
18     }
19     memset(dp, 0x3f3f3f3f, sizeof(dp));
20     dp[0][0] = 0;
21     for(int i = 1; i <= n; i++){
22         for(int j = 0; j <= h; j++){
23             dp[i][j] = min(dp[i-1][j], dp[i][max(0, j
24                 - a[i])] + b[i]);
25         }
26     }
27     cout << dp[n][h] << endl;
28 }

```

11.2 dpd Knapsack 1

```

1  /* dp 背包 - 時間/數量/價值 - 第幾分鐘符合
2  w[i]: 3
3  陣列每一格代表的意義是最大上限為 index
4  時可以放入的最大 value
5  0 0 0 30 30 30 30 30 30
6  w[i]: 4
7  0 0 0 30 50 50 50 80 80
8  w[i]: 5

```

```

8  0 0 0 30 50 60 60 80 90 */
9  int main(){
10     int N, W;
11     cin >> N >> W;
12     int w[10000+5], v[10000+5];
13     for(int i = 0; i < N; i++){
14         cin >> w[i] >> v[i];
15     }
16     long long int dp[10000+5];
17     memset(dp, 0, sizeof(dp));
18     for(int i = 0; i < N; i++){
19         for(int j = W; j >= w[i]; j--){
20             dp[j] = max(dp[j], dp[j - w[i]] + v[i]);
21         }
22     }
23     cout << dp[W] << endl;
24 }

```

11.3 Homer Simpson

```

1  /* dp 背包 - 時間/數量 - 漢堡
2  3 5 54
3  吃 3 分鐘漢堡時
4  0 -1 -1 1 -1 -1 2 -1 -1 3 -1 -1 4 -1 -1 5 -1 -1 6 -1
5  -1 7 -1 -1 8 -1 -1 9 -1 -1 10 -1 -1 11 -1 -1 12
6  -1 -1 13 -1 -1 14 -1 -1 15 -1 -1 16 -1 -1 17 -1
7  -1 18
8  吃 5 分鐘漢堡時 (更新)
9  0 -1 -1 1 -1 1 2 -1 2 3 2 3 4 3 4 5 4 5 6 5 6 7 6 7 8
10  7 8 9 8 9 10 9 10 11 10 11 12 11 12 13 12 13 14
11  13 14 15 14 15 16 15 16 17 16 17 18
12  只有當該時間可剛好吃滿漢堡時會更新
13 全部初始設 -1，用以判斷 譬如當 1 分鐘時
14 吃不了任何漢堡*/
15 int main(){
16     int m, n, t;
17     while(cin >> m >> n >> t){
18         int dp[10000+5];
19         memset(dp, -1, sizeof(dp));
20         dp[0] = 0;
21         for(int i = m; i <= t; i++){
22             if(dp[i - m] != -1)
23                 dp[i] = max(dp[i], dp[i - m] + 1);
24         }
25         for(int i = n; i <= t; i++){
26             if(dp[i - n] != -1)
27                 dp[i] = max(dp[i], dp[i - n] + 1);
28         }
29         // 時間無法剛好吃滿的時候
30         if(dp[t] == -1){
31             for(int i = t; i >= 0; i--){
32                 if(dp[i] != -1){
33                     cout << dp[i] << " " << t - i <<
34                         endl;
35                     break;
36                 }
37             }
38         }
39         else cout << dp[t] << endl;
40     }
41 }

```

11.4 Let Me Count The Ways

```

1  /* dp - 時間/數量 - 硬幣排序
2  要湊出 17
3  1 1 1 1 1 2 2 2 2 2 4 4 4 4 4 6 6 */
4  int main(){
5     long long int n;
6     long long int dp[30000+5];
7     int coin[] = {1, 5, 10, 25, 50};
8     memset(dp, 0, sizeof(dp));
9     // 直接把 dp 做好
10     dp[0] = 1;
11     for(int i = 0; i < 5; i++){
12         for(int j = coin[i]; j < 30000+5; j++){
13             if(dp[j - coin[i]] != -1)
14                 dp[j] += dp[j - coin[i]];
15         }
16     }
17 }

```

```

15 while(cin >> n){
16     if(dp[n] == 1)
17         cout << "There is only " << dp[n] << "
18             way to produce " << n << " cents
19             change." << endl;
20     else
21         cout << "There are " << dp[n] << " ways
22             to produce " << n << " cents change."
23             << endl;
24 }
25 }

```

11.5 Luggage

```

1  /* dp 背包 - 重量/是否成立
2  7 7 13 1
3  1 1 0 0 0 0 0 1 1 0 0 0 0 1 1 1 0 0 0 0 1 1 0 0 0 0 0
4  Note: dp[0] = true */
5  int main(){
6      int t;
7      cin >> t;
8      cin.ignore();
9      while(t--){
10         string str;
11         getline(cin, str);
12         vector<int> v;
13         stringstream ss;
14         int num, cnt = 0, sum = 0;;
15         bool dp[4000+5];
16         memset(dp, false, sizeof(dp));
17         ss << str;
18         while(ss >> num){
19             cnt++;
20             sum += num;
21             v.emplace_back(num);
22         }
23         if(sum & 1){
24             cout << "NO" << endl;
25             continue;
26         }
27         dp[0] = true;
28         for(int i = 0; i < v.size(); i++){
29             for(int j = sum; j >= v[i]; j--){
30                 if(dp[j - v[i]])
31                     dp[j] = true;
32             }
33             cout << (dp[sum/2] ? "YES" : "NO") << endl;
34         }
35     }
36 }

```

11.6 Partitioning by Palindromes

```

1  /* string & dp - 字串長度判斷迴文
2  r a c e c a r
3  i = 0, j = 0
4  -> r = r, dp[1] = dp[0] + 1 = 1
5  i = 1, j = 0
6  -> 因 a != r, dp[2] = 0x3f3f3f3f
7  i = 1, j = 1
8  -> 因 a = a, dp[2] = dp[1] + 1 = 2 */
9  bool check_palindromes(int lef, int rig){
10     // 比較字串兩端都是迴文
11     while(lef < rig){
12         if(str[lef] != str[rig]) return 0;
13         lef++;
14         rig--;
15     }
16     return 1;
17 }
18 int main(){
19     int t;
20     cin >> t;
21     while(t--){

```

```

22     cin >> str;
23     memset(dp, 0x3f3f3f3f, sizeof(dp));
24     dp[0] = 0;
25     for(int i = 0; i < str.size(); ++i)
26         for(int j = 0; j <= i; ++j)
27             if(str[i] == str[j])
28                 if(check_palindromes(j, i))
29                     if(dp[i+1] > dp[j] + 1)
30                         dp[i+1] = dp[j] + 1;
31     cout << dp[str.size()] << endl;
32 }
33 }

```

11.7 SuperSale

```

1  /* dp 背包 - 重量/價值/不可重複使用
2  第一個人的負重: 23
3  0 0 0 0 52 52 52 52 52 54 54 54 54 106 106 106 106
4      106 106 106 106 106 151 151
5  第二個人的負重: 20
6  0 0 0 0 52 52 52 52 52 54 54 54 54 106 106 106 106
7      106 106 106 106
8  第三個人的負重: 20
9  0 0 0 0 52 52 52 52 52 54 54 54 54 106 106 106 106
10     106 106 106 106
11  第四個人的負重: 26
12  0 0 0 0 52 52 52 52 52 54 54 54 54 106 106 106 106
13     106 106 106 106 106 151 151 151 151 151 */
14 struct Edge{
15     int p;
16     int w;
17 }edge[1000+5];
18 int main(){
19     int t;
20     cin >> t;
21     while(t--){
22         int n; cin >> n;
23         for(int i = 0; i < n; i++){
24             cin >> edge[i].p >> edge[i].w;
25         }
26         int g, total = 0;
27         cin >> g;
28         for(int i = 0; i < g; i++){
29             int pw; cin >> pw;
30             int dp[30+5];
31             memset(dp, 0, sizeof(dp));
32             for(int j = 0; j < n; j++){
33                 for(int k = pw; k >= edge[j].w; k--){
34                     dp[k] = max(dp[k], dp[k -
35                         edge[j].w] + edge[j].p);
36                 }
37             }
38             total += dp[pw];
39         }
40         cout << total << endl;
41     }
42 }

```

11.8 Walking on the Safe Side

```

1  /* dp - 地圖更新
2  更新地圖
3  一張如下的地圖 其 dp 更新方法為加上和加左的路
4  0 0 0 0 0
5  0 1 0 0 0
6  0 0 1 0 1
7  0 0 0 0 0
8  1 1 1 1 1
9  1 0 1 2 3
10 1 1 0 2 0
11 1 2 2 4 4 */
12 bool mp[100+5][100+5];
13 long long int dp[100+5][100+5];
14 int main(){
15     int t; cin >> t;
16     bool space = false;

```

```

17 while(t--){
18     if(space) cout << endl;
19     else space = true;
20     int r, c; cin >> r >> c;
21     cin.ignore();
22     memset(mp, false, sizeof(mp));
23     memset(dp, 0, sizeof(dp));
24     string str;
25     for(int i = 0; i < r; i++){
26         getline(cin, str);
27         int n, num;
28         stringstream ss(str);
29         ss >> n;
30         while(ss >> num)
31             mp[n][num] = true;
32     }
33     dp[1][1] = 1;
34     for(int i = 1; i <= r; i++){
35         for(int j = 1; j <= c; j++){
36             if(mp[i][j]) continue;
37             if(i > 1)
38                 dp[i][j] += dp[i-1][j];
39             if(j > 1)
40                 dp[i][j] += dp[i][j-1];
41         }
42     }
43     cout << dp[r][c] << endl;
44 }
45 }

```

11.9 Cutting Sticks

```

1  /* dp - 動態切割取最小
2  100
3  3
4  25 50 75
5  dp:
6  0 0 50 125 200
7  0 0 0 50 125
8  0 0 0 0 50
9  0 0 0 0 0
10 0 0 0 0 0 */
11 int main(){
12     int l;
13     while(cin >> l && l){
14         int n;
15         cin >> n;
16         vector<int> s(n+2);
17         s[0] = 0;
18         for(int i = 1; i <= n; ++i)
19             cin >> s[i];
20         // 從現在開始 n 的數量變為 n + 1
21         s[++n] = 1;
22         int dp[n+5][n+5];
23         memset(dp, 0, sizeof(dp));
24         // r: 切幾段 b: 起點 c: 中間點 e: 終點
25         for(int r = 2; r <= n; ++r){
26             for(int b = 0; b < n; ++b){
27                 // 如果從 b 開始切 r 刀會超出長度就
28                 // break
29                 if(b + r > n) break;
30                 // e: 從 b 開始切 r 刀
31                 int e = b + r;
32                 dp[b][e] = 0x3f3f3f3f;
33                 // c: 遍歷所有從 b 開始到 e
34                 // 結束的中間點
35                 for(int c = b + 1; c < e; ++c){
36                     // dp[b][c] 從 b 到 c 最少 cost +
37                     // dp[c][e] 從 c 到 e 最少 cost
38                     // s[e] - s[b] 兩段之間的 cost
39                     dp[b][e] = min(dp[b][e], dp[b][c]
40                                     + dp[c][e] + s[e] - s[b]);
41                 }
42             }
43         }
44     }
45 }

```

```

40     cout << "The minimum cutting is " << dp[0][n]
41         << "." << endl;
42 }

```

11.10 Race to 1

```

1  /* dp - 數量
2  期望值、質數、dfs */
3  const int N = 1000000;
4  bool sieve[N+5];
5  vector<int> pri;
6  double dp[N+5];
7  // 線性篩
8  void Linear_Sieve(){
9      for (int i = 2; i < N; i++){
10         if (!sieve[i])
11             pri.push_back(i);
12         for (int p: pri){
13             if (i * p >= N) break;
14             sieve[i * p] = true;
15             if (i % p == 0) break;
16         }
17     }
18 }
19 double dfs(int n){
20     if(dp[n] != -1) return dp[n];
21     dp[n] = 0;
22     if(n == 1) return dp[n];
23     int total = 0, prime = 0;
24     for(int i = 0; i < pri.size() && pri[i] <= n;
25         i++){
26         total++;
27         if(n % pri[i]) continue;
28         prime++;
29         dp[n] += dfs(n/pri[i]);
30     }
31     // 算期望值
32     dp[n] = (dp[n] + total)/prime;
33     return dp[n];
34 }
35 int main(){
36     int t, num, ca = 1;
37     for(int i = 0; i <= N; i++)
38         dp[i] = -1;
39     Linear_Sieve();
40     cin >> t;
41     while(t--){
42         cin >> num;
43         cout << "Case " << ca++ << ": " << fixed <<
44             setprecision(10) << dfs(num) << endl;
45     }
46 }

```

11.11 Apple

```

1  /* dp - 數量
2  col = 蘋果 n
3  row = 盤子 m
4  * 0 1 2 3 4
5  1 1 1 1 1
6  2 1 1 2 2 3
7  3 1 1 2 3 4 */
8  int dp[10+5];
9  int main(){
10     int t; cin >> t;
11     while(t--){
12         int n, m;
13         cin >> m >> n;
14         memset(dp, 0, sizeof(dp));
15         dp[0] = 1;
16         for(int i = 1; i <= n; ++i)
17             for(int j = i; j <= m; ++j)

```

```

18     dp[j] += dp[j - i];
19     cout << dp[m] << endl;
20 }
21 }

```

11.12 Stamps

```

1  /* dp - dfs/分配可能性並更新 */
2  const int maxn = 100+5;
3  int h, k, r, maxi = 0;
4  int x[maxn], y[maxn];
5  int ans[maxn]; // 存可貼出最大郵票值的面額
6  void dfs(int i){
7      // 若 x[i] 的 i 多於可貼的郵票數量
8      if(i >= k){
9          if(r > maxi){
10             maxi = max(maxi, r);
11             for(int i = 0; i < k; ++i)
12                 ans[i] = x[i];
13         }
14         return;
15     }
16     // 存此層尚未更新前的 r、y 值，因為 dfs
17     // 完要回去上一層
18     int r_before_this_layer = r;
19     int y_before_this_layer[maxn];
20     for(int j = 0; j < maxn; ++j)
21         y_before_this_layer[j] = y[j];
22     // next: 下一可考慮的郵票面額
23     // postage: 貼完郵票的總面額(y的idx)
24     // num: 要貼幾張
25     // x[i-1] 要 -1 是因為 x 從 0 開始存第一種面額
26     for(int next = x[i-1] + 1; next <= r + 1; ++next){
27         x[i] = next;
28         for(int postage = 0; postage < x[i-1] * h;
29             ++postage){
30             if(y[postage] >= h) continue;
31             for(int num = 1; num <= h - y[postage];
32                 ++num)
33                 if(y[postage] + num < y[postage + num]
34                     * next) && (postage + num * next
35                         < maxn))
36                     y[postage + num * next] =
37                         y[postage] + num;
38         }
39         // 更新現在連續最大值到多少
40         while(y[r+1] < 0x3f3f3f) r++;
41         // x 可貼面額種類多 1
42         dfs(i+1);
43         // 還原 r、y 值
44         r = r_before_this_layer;
45         for(int j = 0; j < maxn; ++j)
46             y[j] = y_before_this_layer[j];
47     }
48 }
49 int main(){
50     while(cin >> h >> k && h && k){
51         memset(x, 0, sizeof(x));
52         memset(y, 0x3f3f3f, sizeof(y));
53         x[0] = 1;
54         r = h;
55         // x[0] = 1, 1 張郵票可貼到的最大值
56         for(int i = 0; i <= r; ++i)
57             y[i] = i;
58         maxi = 0;
59         dfs(1);
60         for(int i = 0; i < k; ++i)
61             printf("%3d", ans[i]);
62         printf(" ->%3d\n", maxi);
63     }
64 }

```

11.13 Evacuation Plan

```

1  /* dp - 路徑/隊伍分配救難所 */
2  const int maxn = 4000+5;
3  int path[maxn][maxn];
4  long long int dp[maxn][maxn];
5  struct Edge{
6      int idx, position;
7      bool operator < (const Edge &rhs) const{
8          return position < rhs.position;
9      }
10 } team[maxn], shelter[maxn];
11 int main(){
12     int n;
13     while(cin >> n){
14         for(int i = 1; i <= n; ++i){
15             cin >> team[i].position;
16             team[i].idx = i;
17         }
18         sort(team + 1, team + n + 1);
19         int m; cin >> m;
20         for(int i = 1; i <= m; ++i){
21             cin >> shelter[i].position;
22             shelter[i].idx = i;
23         }
24         sort(shelter + 1, shelter + m + 1);
25         memset(dp, 0x3f3f3f, sizeof(dp));
26         dp[1][0] = 0;
27         for(int i = 1; i <= m; ++i){
28             for(int j = i; j <= n; ++j){
29                 // dp[i][j] = min(dp[i][j-1],
30                     // dp[i-1][j-1]) +
31                     // abs(team[j].position -
32                         // shelter[i].position);
33                 if(dp[i][j-1] <= dp[i-1][j-1]){
34                     dp[i][j] = min(dp[i][j-1],
35                         dp[i-1][j-1]) +
36                         abs(team[j].position -
37                             shelter[i].position);
38                     path[i][j] = 0; //
39                     // 從左邊來，前面的 teams 有人來
40                     // j shelter
41                 }
42                 else{
43                     dp[i][j] = min(dp[i][j-1],
44                         dp[i-1][j-1]) +
45                         abs(team[j].position -
46                             shelter[i].position);
47                     path[i][j] = 1; //
48                     // 從左上來，前面的 teams 不會來
49                     // j shelter
50                 }
51             }
52         }
53         int now_shelter = m;
54         int ans[maxn];
55         //
56         // 紀錄路徑，若從左邊來，上一隊也來此；若從右邊來，上
57         for(int i = n; i > 0; --i){
58             ans[team[i].idx] =
59                 shelter[now_shelter].idx;
60             now_shelter -= path[now_shelter][i];
61         }
62         cout << dp[m][n] << endl;
63         for(int i = 1; i < n; ++i)
64             cout << ans[i] << " ";
65         cout << ans[n] << endl;
66     }
67 }

```

11.14 Ladies Choice

```

1  /* dp - ladies & men */
2  const int maxn = 1000+5;
3  int n;

```



```

4 int man[maxn][maxn], manidx[maxn], lady[maxn][maxn],
  ladyidx[maxn];
5 int dp[maxn];
6 deque<int> dq;
7 void dp_func(){
8     while(!dq.empty()){
9         int man_now = dq.front();
10        dq.pop_front();
11        // manidx 現在指著的 lady
12        int lady1 = manidx[man_now];
13        // man 目前最想要的 lady
14        int lady_first = man[man_now][lady1];
15        // ladyidx 現在指著的 man
16        int man1 = ladyidx[lady_first];
17        if(man1 == 0){
18            dp[man_now] = lady_first;
19            ladyidx[lady_first] = man_now;
20        }
21        else if(lady[lady_first][man1] >
22            lady[lady_first][man_now]){
23            dp[man_now] = lady_first;
24            manidx[man1]++;
25            dq.emplace_back(man1);
26            ladyidx[lady_first] = man_now;
27        }
28        else{
29            dq.emplace_back(man_now);
30            manidx[man_now]++;
31        }
32    }
33 int main(){
34     int t; cin >> t;
35     bool space = false;
36     while(t--){
37         cin >> n;
38         if(space) cout << endl;
39         space = true;
40         memset(man, 0, sizeof(man));
41         memset(lady, 0, sizeof(lady));
42         memset(manidx, 0, sizeof(manidx));
43         memset(ladyidx, 0, sizeof(ladyidx));
44         dq.clear();
45         for(int i = 1; i <= n; ++i){
46             for(int j = 1; j <= n; ++j)
47                 cin >> man[i][j];
48             dq.emplace_back(i);
49             manidx[i] = 1;
50         }
51         for(int i = 1; i <= n; ++i){
52             for(int j = 1; j <= n; ++j){
53                 int man_lady;
54                 cin >> man_lady;
55                 lady[i][man_lady] = j;
56             }
57         }
58         dp_func();
59         for(int i = 1; i <= n; ++i)
60             cout << dp[i] << endl;
61     }
62 }

```

12 LIS

12.1 Wavio Sequence

```

1 /* LIS、LDS */
2 int N;
3 const int maxn = 10000 + 5;
4 int length[maxn];
5 int seq[maxn], revseq[maxn];
6 void LIS(vector<int> &s){
7     if(s.size() == 0) return;
8     vector<int> v;

```

```

9     v.emplace_back(s[0]);
10    seq[0] = 1;
11    for(int i = 1; i < s.size(); ++i){
12        int n = s[i];
13        if(n > v.back())
14            v.push_back(n);
15        else
16            *lower_bound(v.begin(), v.end(), n) = n;
17        seq[i] = v.size();
18    }
19    return;
20 }
21 void LDS(vector<int> &s){
22     if(s.size() == 0) return;
23     vector<int> v;
24     v.emplace_back(s[0]);
25     revseq[0] = 1;
26     for(int i = 1; i < s.size(); ++i){
27         int n = s[i];
28         if(n > v.back())
29             v.push_back(n);
30         else
31             *lower_bound(v.begin(), v.end(), n) = n;
32         revseq[i] = v.size();
33     }
34     return;
35 }
36 int main(){
37     while(cin >> N){
38         vector<int> s(N), revs(N);
39         for(int i = 0; i < N; i++){
40             cin >> s[i];
41             revs[i] = s[i];
42         }
43         reverse(revs.begin(), revs.end());
44         LIS(s);
45         LDS(revs);
46         reverse(revseq, revseq + N);
47         int maxi = -1;
48         for(int i = 0; i < N; i++){
49             if(min(seq[i], revseq[i]) > maxi)
50                 maxi = min(seq[i], revseq[i]);
51         }
52         cout << maxi * 2 - 1 << endl;
53     }

```

12.2 Robots II

```

1 /* LIS
2 No.: 2 4 11 13 25 28 41 42
3 LIS: 1 2 3 4 4 5 5 5
4 num: 1 1 1 1 1 2 2 4
5 path: -1 0 1 2 2 3 3 5 */
6 const int maxn = 100+5;
7 int r, c;
8 vector<int> G;
9 int LIS[maxn * maxn], num[maxn * maxn], path[maxn *
  maxn];
10 bool garbage[maxn][maxn];
11 void show_path(int n){
12     if(path[n] != -1) show_path(path[n]);
13     if((n != G.size() - 1) || garbage[r][c]) cout <<
14         " " << G[n];
15 }
16 int main(){
17     int ca = 1;
18     while(cin >> r >> c && (r != -1) && (c != -1)){
19         memset(garbage, false, sizeof(garbage));
20         G.clear();
21         int x, y;
22         while(cin >> x >> y && x && y){
23             garbage[x][y] = true;
24         }
25         // 紀錄有垃圾的點的編號
26         for(int i = 1; i <= r; ++i){

```



```

26     for(int j = 1; j <= c; ++j){
27         if(garbage[i][j]) G.emplace_back((i -
28             1) * c + j);
29     }
30     // 如果終點沒有垃圾，假設他有
31     if(!garbage[r][c]) G.emplace_back(r * c);
32     G.emplace_back(0);
33     // i 和 j
34     // 是按照編號大小順序由小排到大的垃圾編號
35     for(int i = 0; i < G.size(); ++i){
36         LIS[i] = 1;
37         num[i] = 1;
38         path[i] = -1;
39         for(int j = 0; j < i; ++j){
40             // 判斷垃圾的 col 前後
41             if(((G[j] - 1) % c) <= ((G[i] - 1) %
42                 c)){
43                 // num 是經過的路徑數量。path
44                 // 是從誰來
45                 if(LIS[i] == LIS[j] + 1){
46                     num[i] += num[j];
47                 }
48                 else if(LIS[i] < LIS[j] + 1){
49                     LIS[i] = LIS[j] + 1;
50                     num[i] = num[j];
51                     path[i] = j;
52                 }
53             }
54         }
55         G.pop_back();
56         // 要把假設還回去
57         if(!garbage[r][c]) LIS[G.size() - 1]--;
58         cout << "CASE#" << ca++ << ": " <<
59             LIS[G.size() - 1] << " " << num[G.size()
60             - 1];
61         show_path(G.size() - 1);
62         cout << endl;
63     }
64 }

```

13 Math

13.1 Big Mod

```

1  '''
2  Mod
3  pow(x, y, z) = x^y % z
4  '''
5  # python 如何讀取直到 EOF 用 try except
6  try:
7      while True:
8          # input().split() 用空格切開讀取一整行
9          # map (型態, input().split()) 才能把值全讀成
10             int
11             B, P, M = map(int, input().split())
12             print(pow(B, P, M))
13 except EOFError:
14     exit

```

13.2 Bubble Sort Expect Value

```

1  /* 數論 期望值算法:
2  擲一枚公平的六面骰子，其每次「點數」的期望值是 3.5
3   $E(x) = 1 * 1/6 + 2 * 1/6 + 3 * 1/6 + 4 * 1/6 + 5 * 1/6 + 6 * 1/6$ 
4   $= (1 + 2 + 3 + 4 + 5 + 6)/6 = 3.5$ 
5  bubble sort 每兩兩之間交換機率是 1/2
6  總共會做  $C(n, 2)$  次
7   $E(x) = C(n, 2) * 1/2 = (n * (n - 1))/2 * 1/2 *$ 

```

```

8  int t, ca = 1;
9  cin >> t;
10 while(t--){
11     long long int n;
12     cin >> n;
13     cout << "Case " << ca++ << ": ";
14     // 如果  $(n * (n - 1))$  可以被 4 整除
15     // 代表最後答案會是整數，否則會是分數
16     if((n * (n - 1)) % 4){
17         cout << ( (n * (n - 1)) / 2 ) << "/" << endl;
18     }
19     else{
20         cout << ( (n * (n - 1)) / 2 ) / 2 << endl;
21     }
22 }

```

13.3 Fraction Floor Sum

```

1  /* 數論
2   $[N/i] == M$ 
3   $\rightarrow M \leq N/i < M + 1$ 
4   $\rightarrow N/(M+1) < i \leq N/M$ 
5  int main(){
6     long long int N;
7     cin >> N;
8     long long int ans = 0;
9     for(long long int i = 1; i <= N; i++){
10         long long int M = N / i, n = N / M;
11         // 總共會有  $n - i$  個的  $[N/i]$  值都是  $M$ 
12         ans += (n - i + 1) * M;
13         // 更新跳過 以免重複計算
14         i = n;
15     }
16     cout << ans << endl;
17 }

```

13.4 How Many 0s

```

1  /* 數論 */
2  int main(){
3     long long int n, m;
4     while(cin >> n >> m && (n >= 0) && (m >= 0)){
5         long long int total1 = 0, total2 = 0;
6         long long int ten = 1, tmp = n-1;
7         while(tmp >= 10){
8             if(tmp % 10 == 0){
9                 tmp /= 10;
10                total1 += (tmp - 1) * ten + ((n-1) %
11                    ten) + 1;
12            }
13            else{
14                tmp /= 10;
15                total1 += tmp * ten;
16            }
17            ten *= 10;
18        }
19        ten = 1; tmp = m;
20        while(tmp >= 10){
21            if(tmp % 10 == 0){
22                tmp /= 10;
23                total2 += (tmp - 1) * ten + (m % ten)
24                    + 1;
25            }
26            else{
27                tmp /= 10;
28                total2 += tmp * ten;
29            }
30            ten *= 10;
31        }
32        if(n == 0) total1--;
33        cout << total2 - total1 << endl;
34    }
35 }

```

13.5 Number of Pairs

```

1  /* 數論
2  upper_bound ex:
3  10 20 30 30 40 50
4  upper_bound for element 30 is at index 4
5  lower_bound ex:
6  10 20 30 40 50
7  lower_bound for element 30 at index 2 */
8  int main(){
9      int t;
10     cin >> t;
11     while(t--){
12         int n, l, r;
13         vector<int> v;
14         cin >> n >> l >> r;
15         int num;
16         for(int i = 0; i < n; i++){
17             cin >> num;
18             v.emplace_back(num);
19         }
20         sort(v.begin(), v.end());
21         long long int ans = 0;
22         for(int i = 0; i < n; i++){
23             ans += (upper_bound(v.begin() + i + 1,
24                               v.end(), r - v[i]) -
25                    lower_bound(v.begin() + i + 1,
26                               v.end(), l - v[i]));
27         }
28         cout << ans << endl;
29     }
30 }

```

13.6 ORXOR

```

1  /* bitwise operator 二進位制數論
2  如何切區段，之所以要1<<n是為了可以跑000~111
3  i = 0, binary i = 000
4  0 : 1 5 7
5  i = 1, binary i = 001
6  1 : 1 5 7
7  i = 2, binary i = 010，看得出來切了一刀
8  2 : 1 | 5 7
9  i = 3, binary i = 011
10 3 : 1 | 5 7
11 i = 4, binary i = 100，為了要切在index=2，所以才要1<<j
12 4 : 1 5 | 7
13 i = 5, binary i = 101
14 5 : 1 5 | 7
15 i = 6, binary i = 110
16 6 : 1 | 5 | 7
17 i = 7, binary i = 111
18 7 : 1 | 5 | 7
19 可以觀察出來，前兩位 bit 是 1 時代表的意義是切在哪裡
20 */
21 int main(){
22     int n; cin >> n;
23     int num[20+7];
24     memset(num, 0, sizeof(num));
25     for(int i = 1; i <= n; i++){
26         cin >> num[i];
27         // 不知道為甚麼只有 2147483647 給過
28         int mini = 2147483647;
29         // 1 << n = n * 2
30         for(int i = 0; i < (1 << n); i++){
31             int XOR = 0, OR = 0;
32             for(int j = 1; j <= n; j++){
33                 OR |= num[j];
34                 if((i & (1 << j))){
35                     XOR ^= OR;
36                     OR = 0;
37                 }
38             }
39             XOR ^= OR;
40             mini = min(mini, XOR);

```

```

40     }
41     cout << mini << endl;
42 }

```

13.7 X drawing

```

1  /* 數論畫圖 */
2  int main(){
3      long long int n;
4      long long int a, b;
5      long long int p, q, r, s;
6      cin >> n >> a >> b;
7      cin >> p >> q >> r >> s;
8      for(long long int i = p; i <= q; i++){
9          for(long long int j = r; j <= s; j++){
10             if(abs(i - a) == abs(j - b)) cout << '#';
11             else cout << '.';
12         }
13         cout << endl;
14     }

```

13.8 Playing With Stones

```

1  /* Nim Game - SG 函數 */
2  long long int SG(long long int n){
3      return n % 2 == 0 ? n/2 : SG(n/2);
4  }
5  int main(){
6      int t;
7      cin >> t;
8      while(t--){
9          int n;
10         cin >> n;
11         long long int a, v = 0;
12         for(int i = 0; i < n; ++i){
13             cin >> a;
14             v ^= SG(a);
15         }
16         if(v) cout << "YES" << endl;
17         else cout << "NO" << endl;
18     }
19 }

```

13.9 And Then There Was One

```

1  /* 環狀取石頭更新
2  f(1)=0
3  f(i)=(f(i-1)+k)%i
4  f(n)=(f(n-1)+m)%n
5  最後石頭編號: f(n)+1=1 */
6  const int maxn = 10000+5;
7  int f[maxn];
8  int main(){
9      int n, k, m;
10     while(cin >> n >> k >> m && n && k && m){
11         f[1] = 0;
12         // i 是剩下的石頭數量
13         for(int i = 2; i < n; ++i){
14             f[i] = (f[i-1] + k) % i;
15         }
16         f[n] = (f[n-1] + m) % n;
17         cout << f[n] + 1 << endl;
18     }
19 }

```

14 Binary Search

14.1 Fill the Containers

```

1  /*binary_search 變形*/
2  int binary_search(int arr[maxn], int lef, int rig,
3      int mini){
4      if(lef > rig) return mini;
5      int amount = 1, fill = 0;
6      int mid = (lef + rig) >> 1;
7      for(int i = 0; i < n; ++i){
8          if(amount > m) break;
9          fill += arr[i];
10         if(fill > mid){
11             fill = arr[i];
12             amount++;
13         }
14     }
15     if(!flag && amount <= m) mini = mid;
16     if(flag && amount == m) mini = mid;
17     if(amount == m){
18         flag = true;
19         return binary_search(arr, lef, mid - 1, mid);
20     }
21     else if(amount < m){
22         return binary_search(arr, lef, mid - 1, mini);
23     }
24     else{
25         return binary_search(arr, mid + 1, rig, mini);
26     }
27 }
28 int main(){
29     int ca = 1;
30     while(cin >> n >> m){
31         flag = false;
32         int arr[maxn];
33         int maxi = 0, sum = 0;
34         for(int i = 0; i < n; ++i){
35             cin >> arr[i];
36             sum += arr[i];
37             maxi = max(maxi, arr[i]);
38         }
39         cout << binary_search(arr, maxi, sum, maxi)
40             << endl;
41     }
42 }

```

14.2 Where is the marble

```

1  /*upper_bound & lower_bound*/
2  int main(){
3      int N, Q;
4      int ca = 1;
5      while(cin >> N >> Q && N && Q){
6          vector<int> v(N);
7          for(int i = 0; i < N; ++i) cin >> v[i];
8          sort(v.begin(), v.end());
9          cout << "CASE# " << ca++ << " " << endl;
10         int marble;
11         for(int i = 0; i < Q; ++i){
12             cin >> marble;
13             int lef = lower_bound(v.begin(), v.end(),
14                 marble) - v.begin();
15             int rig = upper_bound(v.begin(), v.end(),
16                 marble) - v.begin();
17             if(lef == rig) cout << marble << " not
18                 found" << endl;
19             else{
20                 cout << marble << " found at " << lef
21                     + 1 << endl;
22             }
23         }
24     }
25 }

```

15 Graph

15.1 Maximum sum on a torus

```

1  /* Prefix sum in Graph*/
2  const int maxn = 80;
3  const int inf = 0x3f3f3f3f;
4  int arr[maxn*2 + 5][maxn*2 + 5];
5  int prefix_sum[maxn*2 + 5][maxn*2 + 5];
6  int ans[maxn*2];
7  int n;
8  int maxSub(int start){
9      int maxi, dp;
10     maxi = dp = ans[start];
11     for(int i = start + 1; i < start + n; ++i){
12         dp += ans[i];
13         maxi = max(maxi, dp);
14     }
15     return maxi;
16 }
17 int main(){
18     int t;
19     cin >> t;
20     while(t--){
21         memset(arr, 0, sizeof(arr));
22         cin >> n;
23         for(int i = 0; i < n; ++i)
24             for(int j = 0; j < n; ++j){
25                 cin >> arr[i][j];
26                 arr[n+i][j] = arr[i][n+j] =
27                     arr[n+i][n+j] = arr[i][j];
28             }
29         int len = 2*n;
30         memset(prefix_sum, 0, sizeof(prefix_sum));
31         for(int i = 0; i < len; ++i)
32             for(int j = 0; j < len; ++j){
33                 if(i == 0) prefix_sum[i][j] =
34                     arr[i][j];
35                 else prefix_sum[i][j] =
36                     prefix_sum[i-1][j] + arr[i][j];
37             }
38         int maxi = -inf;
39         for(int i = 0; i < len; ++i){
40             for(int j = i; j < i + n && j < len; ++j){
41                 for(int k = 0; k < len; ++k){
42                     if(i == 0) ans[k] =
43                         prefix_sum[j][k];
44                     else ans[k] = prefix_sum[j][k] -
45                         prefix_sum[i-1][k];
46                 }
47                 for(int k = 0; k < n; ++k){
48                     int answer = maxSub(k);
49                     maxi = max(maxi, answer);
50                 }
51             }
52         }
53         cout << maxi << endl;
54     }
55 }

```

16 Segement Tree

16.1 Frequent values

```

1  /* Segement Tree & RMQ (Range Sum Query)
2  idx:  1  2  3  4  5  6  7  8  9  10
3  num: -1 -1  1  1  1  1  3  10 10 10
4  fre:  2  2  4  4  4  4  1  3  3  3
5  border
6  left: 1  1  3  3  3  3  7  8  8  8
7  right:2  2  6  6  6  6  7 10 10 10 */
8  #define Lson(x) x << 1
9  #define Rson(x) (x << 1) + 1

```

```

10 const int maxn = 1e5+5;
11 struct Tree{
12     int lef, rig, value;
13 }tree[4 * maxn];
14 struct Num{
15     int lef, rig, value, fre;
16 }num[maxn];
17 // 建立 segement tree
18 void build(int lef, int rig, int x){
19     tree[x].lef = lef;
20     tree[x].rig = rig;
21     // 區塊有多長，題目詢問的重點
22     if(lef == rig){
23         tree[x].value = num[lef].fre;
24         return;
25     }
26     int mid = (lef + rig) >> 1;
27     build(lef, mid, Lson(x));
28     build(mid + 1, rig, Rson(x));
29     tree[x].value = max(tree[Lson(x)].value,
30         tree[Rson(x)].value);
31 }
32 // 查詢 segement tree
33 int query(int lef, int rig, int x){
34     // 題目所查詢的區間剛好在同個區塊上，num[lef].v
35     // == num[rig].v
36     if(num[lef].value == num[rig].value) return rig -
37         lef + 1;
38     int ans = 0;
39     // 查詢的左區間邊界切到區塊，且此區間有數個區塊
40     if(lef > num[lef].lef){
41         // 計算切到的區間大小
42         ans = num[lef].rig - lef + 1;
43         //
44         // 更新左邊界至被切區塊的右邊界加一，就不會切到區
45         lef = num[lef].rig + 1;
46     }
47     // 查詢的右區間邊界切到區塊，且此區間有數個區塊
48     if(rig < num[rig].rig){
49         // 計算切到的區間大小，並找出最大
50         ans = max(ans, rig - num[rig].lef + 1);
51         // 更新右邊界
52         rig = num[rig].lef - 1;
53     }
54     //
55     // 如果左邊界大於右邊界，表示不需要再進行查詢直接回傳
56     if(lef > rig) return ans;
57     if(tree[x].lef >= lef && tree[x].rig <= rig)
58         return tree[x].value;
59     int mid = (tree[x].lef + tree[x].rig) >> 1;
60     if(lef <= mid) ans = max(ans, query(lef, rig,
61         Lson(x)));
62     if(mid < rig) ans = max(ans, query(lef, rig,
63         Rson(x)));
64     return ans;
65 }
66 int main(){
67     int n, q;
68     while(cin >> n && n){
69         cin >> q;
70         int start = 1;
71         for(int i = 1; i <= n; ++i){
72             cin >> num[i].value;
73             if(num[i].value != num[i-1].value){
74                 for(int j = start; j < i; ++j){
75                     num[j].rig = i - 1;
76                     num[j].fre = i - start;
77                 }
78                 start = num[i].lef = i;
79             }
80             else num[i].lef = start;
81         }
82         // 最後一段 [start, n]
83         for(int j = start; j <= n; ++j){
84             num[j].rig = n;
85             num[j].fre = n - start + 1;

```

```

78     }
79     build(1, n, 1);
80     int lef, rig;
81     for(int i = 0; i < q; ++i){
82         cin >> lef >> rig;
83         cout << query(lef, rig, 1) << endl;
84     }
85 }
86 }

```

17 Dijkstra

17.1 Airport Express

```

1 /* Dijkstra 捷徑票 */
2 int n, m, S, T;
3 const int inf = 1e9;
4 const int maxn = 20000 + 5;
5 struct Edge{
6     int v, w;
7 };
8 struct Item{
9     int u, dis;
10    // 取路徑最短
11    bool operator < (const Item &other) const{
12        return dis > other.dis;
13    }
14 };
15 int dis[maxn], from[maxn];
16 vector<Edge> G[maxn];
17 void dijkstra(int s){
18     for(int i = 0; i <= n; i++){
19         dis[i] = inf;
20     }
21     dis[s] = 0;
22     for(int i = 0; i <= n; i++){
23         from[i] = i;
24     }
25     priority_queue<Item> pq;
26     pq.push({s, 0});
27     while(!pq.empty()){
28         // 取路徑最短的點
29         Item now = pq.top();
30         pq.pop();
31         if(now.dis > dis[now.u])
32             continue;
33         // 鬆弛 更新
34         // 把與 now.u 相連的點都跑一遍
35         for(Edge e : G[now.u]){
36             if(dis[e.v] > now.dis + e.w){
37                 dis[e.v] = now.dis + e.w;
38                 from[e.v] = now.u;
39                 pq.push({e.v, dis[e.v]});
40             }
41         }
42     }
43     deque<int> ans;
44     void dfs(int T){
45         ans.emplace_back(T);
46         if(from[T] != T) dfs(from[T]);
47     }
48     int main(){
49         bool space = false;
50         while(cin >> n >> S >> T){
51             if(!space) space = true;
52             else cout << endl;
53             for(int i = 0; i <= n; i++){
54                 G[i].clear();
55             }
56             ans.clear();
57             cin >> m;
58             int u, v, w;
59             for(int i = 0; i < m; i++){
60                 cin >> u >> v >> w;
61                 // 無向圖
62                 G[u].push_back({v, w});

```

```

61         G[v].push_back({u, w});
62     }
63     dijkstra(S);
64     dfs(T);
65     int ori = dis[T];
66     int mini = dis[T], state = 0;
67     int ticket;
68     cin >> ticket;
69     for(int i = 0; i < ticket; ++i){
70         cin >> u >> v >> w;
71         G[u].push_back({v, w});
72         dijkstra(S);
73         if(dis[T] < mini){
74             mini = min(mini, dis[T]);
75             state = u;
76             ans.clear();
77             dfs(T);
78         }
79         G[u].pop_back();
80         G[v].push_back({u, w});
81         dijkstra(S);
82         if(dis[T] < mini){
83             mini = min(mini, dis[T]);
84             state = v;
85             ans.clear();
86             dfs(T);
87         }
88         G[v].pop_back();
89     }
90     for(int i = ans.size()-1; i > 0; i--){
91         cout << ans[i] << " ";
92     }
93     cout << ans[0];
94     cout << endl;
95     if(mini == ori)
96         cout << "Ticket Not Used" << endl;
97     else
98         cout << state << endl;
99     cout << mini << endl;
100 }

```

17.2 Walk Through the Forest

```

1  /* Dijkstra + 路徑最優化 DP */
2  const int inf = 0x3f3f3f3f;
3  const int maxn = 1000+5;
4  int n, m;
5  struct Edge{
6      int v, w;
7  };
8  struct Item{
9      int u, dis;
10     bool operator < (const Item &other) const{
11         return dis > other.dis;
12     }
13 };
14 int dis[maxn];
15 long long int dp[maxn];
16 vector<Edge> G[maxn];
17 vector<int> path[maxn];
18 void dijkstra(int s){
19     for(int i = 0; i <= n; ++i){
20         dis[i] = inf;
21     }
22     dis[s] = 0;
23     priority_queue<Item> pq;
24     pq.push({s, 0});
25     while(!pq.empty()){
26         Item now = pq.top();
27         pq.pop();
28
29         if(now.dis > dis[now.u]){
30             continue;
31         }
32
33         for(Edge e: G[now.u]){

```

```

34             if(dis[e.v] > now.dis + e.w){
35                 dis[e.v] = now.dis + e.w;
36                 pq.push({e.v, dis[e.v]});
37             }
38         }
39     }
40 }
41 long long int dfs(int u){
42     // ans 是 pointer, 指向 dp[u] 的記憶體位址
43     // 對於 ans 的 value 改變會記錄在 dp[u]
44     long long int& ans = dp[u];
45     if(ans != -1) return ans;
46     if(u == 2) return ans = 1;
47     ans = 0;
48     for(int i = 0; i < path[u].size(); ++i)
49         ans += dfs(path[u][i]);
50     return ans;
51 }
52 int main(){
53     while(cin >> n && n){
54         cin >> m;
55         for(int i = 0; i <= n; ++i) G[i].clear();
56         int u, v, w;
57         for(int i = 0; i < m; ++i){
58             cin >> u >> v >> w;
59             G[u].push_back({v, w});
60             G[v].push_back({u, w});
61         }
62         dijkstra(2); // dijkstra
63         // 紀錄從終點到每個點的距離
64         memset(dp, -1, sizeof(dp));
65         for(int i = 1; i <= n; ++i){
66             path[i].clear();
67             for(int j = 0; j < G[i].size(); ++j){
68                 int v = G[i][j].v;
69                 // 如果到 v 的距離比到 i
70                 // 遠, 代表從起點經過 i 再到 v
71                 if(dis[i] > dis[v])
72                     path[i].push_back(v);
73             }
74             cout << dfs(1) << endl;
75         }
76     }
77 }

```

18 Kruskal

18.1 Qin Shi Huang Road System

```

1  /* kruskal disjoint set dfs */
2  const int maxn = 1000 + 5;
3  int n, m;
4  int x[maxn], y[maxn], p[maxn];
5  struct Edge{
6      int u, v;
7      double w;
8      bool operator < (const Edge &rhs) const{
9          return w < rhs.w;
10     }
11 }edge[maxn * maxn];
12 vector<Edge> G[maxn];
13 int parent[maxn];
14 // 計算兩點之間的距離
15 double dist(int a, int b){
16     double x2 = (x[a] - x[b]) * (x[a] - x[b]);
17     double y2 = (y[a] - y[b]) * (y[a] - y[b]);
18     return sqrt(x2 + y2);
19 }
20 // disjoint set
21 int find(int x){
22     return x == parent[x] ? x : parent[x] =
23         find(parent[x]);
24 }
25 bool unite(int a, int b){

```

```

25     int x = find(a);
26     int y = find(b);
27     if(x == y) return false;
28     parent[x] = y;
29     return true;
30 }
31 double kruskal(){
32     m = 0; // m: 邊的數量
33     for(int i = 0; i < n; ++i)
34         for(int j = i + 1; j < n; ++j)
35             edge[m++] = (Edge){i, j, dist(i, j)};
36     sort(edge, edge + m);
37     for(int i = 0; i < n; ++i){
38         parent[i] = i;
39         G[i].clear();
40     }
41     double total = 0.0;
42     int edge_cnt = 0;
43     for(int i = 0; i < m; ++i){
44         int u = edge[i].u, v = edge[i].v;
45         double cnt = edge[i].w;
46         if(unite(u, v)){
47             G[u].push_back((Edge){u, v, cnt});
48             G[v].push_back((Edge){v, u, cnt});
49             total += cnt;
50             if(++edge_cnt == n-1) break;
51         }
52     }
53     return total;
54 }
55 double maxcost[maxn][maxn];
56 bool visited[maxn];
57 void dfs(int u){
58     visited[u] = true;
59     for(int i = 0; i < G[u].size(); ++i){
60         int v = G[u][i].v;
61         if(visited[v]) continue;
62         double cost = G[u][i].w;
63         maxcost[u][v] = maxcost[v][u] = cost;
64         // 更新 MST 樹上的點到 v 點的距離
65         for(int j = 0; j < n; ++j)
66             if(visited[j])
67                 maxcost[j][v] = maxcost[v][j] =
68                     max(maxcost[j][u], cost);
69     }
70 }
71 void solve(){
72     double total = kruskal();
73     memset(maxcost, 0, sizeof(maxcost));
74     memset(visited, false, sizeof(visited));
75     dfs(0);
76     double ans = -1;
77     // 把所有點都遍歷一次
78     for(int i = 0; i < n; ++i)
79         for(int j = i + 1; j < n; ++j)
80             ans = max(ans, (p[i] + p[j]) / (total -
81                 maxcost[i][j]));
82     printf("%.2lf\n", ans);
83 }
84 int main(){
85     int t;
86     scanf("%d", &t);
87     while(t--){
88         scanf("%d", &n);
89         for(int i = 0; i < n; ++i)
90             scanf("%d%d%d", &x[i], &y[i], &p[i]);
91         solve();
92     }
93     return 0;
94 }

```

19 Bipartite Graph

19.1 Claw Decomposition

```

1  /*二分圖 Bipatirate*/
2  const int maxn = 300+5;
3  int n;
4  int color[maxn];
5  vector<vector<int>> v(maxn);
6  bool dfs(int s){
7      for(auto it : v[s]){
8          if(color[it] == -1){
9              //
10             // 如果與點相連又還未填色，填塞成與原點不同的另一色
11             color[it] = 3 - color[s];
12             // 同樣對此點去判定與此點相連的點的填色
13             if(!dfs(it)) return false;
14         }
15         if(color[s] == color[it]){
16             // 如果相鄰兩點同色，回傳 false
17             return false;
18         }
19     }
20     return true;
21 }
22 void isBipatirate(){
23     bool flag = true;
24     for(int i = 1; i <= n; ++i){
25         if(color[i] == -1){
26             // 如果還未填色過，就先填色成
27             // 1，並對與此點相連的點都 dfs 判定填色
28             color[i] = 1;
29             flag &= dfs(i);
30         }
31     }
32     if(flag) cout << "YES" << endl;
33     else cout << "NO" << endl;
34 }
35 int main(){
36     while(cin >> n && n){
37         for(int i = 1; i <= n; ++i) v[i].clear();
38         memset(color, -1, sizeof(color));
39         int a, b;
40         while(cin >> a >> b && (a || b)){
41             v[a].emplace_back(b);
42             v[b].emplace_back(a);
43         }
44         isBipatirate();
45     }
46 }

```

19.2 Guardian of Decency

```

1  /* 二分圖最大匹配
2  匈牙利演算法 Hungarian algorithm*/
3  const int maxn = 500+5;
4  int bn, gn;
5  int match[maxn];
6  bool visited[maxn];
7  vector<vector<int>> G(maxn);
8  struct People{
9      int h;
10     string music, sport;
11     // constructor
12     People(){
13         People(int h, string music, string sport){
14             this->h = h;
15             this->music = music;
16             this->sport = sport;
17         }
18     }lef[maxn], rig[maxn];
19     bool check(People boy, People girl){

```

```

20     if(abs(boy.h - girl.h) <= 40 && boy.music ==
        girl.music && boy.sport != girl.sport) return
        true;
21     return false;
22 }
23 bool dfs(int s){
24     for(int i = 0; i < G[s].size(); ++i){
25         int v = G[s][i];
26         if(visited[v]) continue;
27         visited[v] = true;
28         // 如果這個女生還沒被配對過，直接匹配
29         // 如果已經被配對，則根據這個女生所配對的對象
            dfs 重新匹配所有人的對象
30         if(match[v] == -1 || dfs(match[v])){
31             match[v] = s;
32             return true;
33         }
34     }
35     return false;
36 }
37 int Hungarian(){
38     int cnt = 0;
39     memset(match, -1, sizeof(match));
40     for(int i = 0; i < bn; ++i){
41         memset(visited, false, sizeof(visited));
42         if(dfs(i)) cnt++;
43     }
44     return cnt;
45 }
46 int main(){
47     int t;
48     cin >> t;
49     while(t--){
50         int N;
51         cin >> N;
52         bn = 0, gn = 0;
53         for(int i = 0; i <= N; ++i) G[i].clear();
54         int h;
55         string sex, music, sport;
56         for(int i = 0; i < N; ++i){
57             cin >> h >> sex >> music >> sport;
58             if(sex == "M")
59                 lef[bn++] = People(h, music, sport);
60             else
61                 rig[gn++] = People(h, music, sport);
62         }
63         for(int i = 0; i < bn; ++i)
64             for(int j = 0; j < gn; ++j)
65                 if(check(lef[i], rig[j]))
66                     G[i].emplace_back(j);
67         cout << N - Hungarian() << endl;
68     }
69 }

```

19.3 Taxi Cab Scheme

```

1  /* 二分圖最大匹配
2  匈牙利演算法 Hungarian algorithm */
3  const int maxn = 500+5;
4  int n;
5  int match[maxn];
6  bool visited[maxn];
7  vector<int> G[maxn];
8  struct People{
9      int s, x1, y1, x2, y2;
10     bool operator < (const People & rhs) const {
11         return s < rhs.s;
12     }
13 }p[maxn];
14 bool check(People boy, People girl){
15     int tmp = boy.s + abs(boy.x2 - boy.x1) +
        abs(boy.y2 - boy.y1) + abs(boy.x2 - girl.x1)
        + abs(boy.y2 - girl.y1);
16     if(tmp < girl.s) return true;
17     return false;

```

```

18 }
19 bool dfs(int s){
20     for(int i = 0; i < G[s].size(); ++i){
21         int v = G[s][i];
22         if(visited[v]) continue;
23         visited[v] = true;
24         if(match[v] == -1 || dfs(match[v])){
25             match[v] = s;
26             return true;
27         }
28     }
29     return false;
30 }
31 int Hungarian(){
32     int cnt = 0;
33     memset(match, -1, sizeof(match));
34     for(int i = 0; i < n; ++i){
35         memset(visited, false, sizeof(visited));
36         if(dfs(i)) cnt++;
37     }
38     return cnt;
39 }
40 int main(){
41     int t;
42     scanf("%d", &t);
43     while(t--){
44         scanf("%d", &n);
45         for(int i = 0; i < n; ++i) G[i].clear();
46         for(int i = 0; i < n; ++i){
47             int h, m;
48             scanf("%d:%d", &h, &m);
49             p[i].s = h * 60 + m;
50             scanf("%d%d%d%d", &p[i].x1, &p[i].y1,
                    &p[i].x2, &p[i].y2);
51         }
52         sort(p, p + n);
53         for(int i = 0; i < n; ++i)
54             for(int j = i + 1; j < n; ++j)
55                 if(check(p[i], p[j]))
56                     G[i].push_back(j);
57         printf("%d\n", n - Hungarian());
58     }
59 }

```

19.4 SAM I AM

```

1  /* 二分圖匹配 + 最小點覆蓋 */
2  const int maxn = 1000+5;
3  int R, C, N;
4  bool arr[maxn][maxn], visitX[maxn], visitY[maxn];
5  int matchX[maxn], matchY[maxn];
6  int dfs(int x){
7      visitX[x] = true;
8      for(int y = 1; y <= C; ++y){
9          if(arr[x][y] && !visitY[y]){
10             visitY[y] = true;
11             if(matchY[y] == 0 || dfs(matchY[y])){
12                 matchX[x] = y;
13                 matchY[y] = x;
14                 return 1;
15             }
16         }
17     }
18     return 0;
19 }
20 int Match(){
21     int sum = 0;
22     memset(matchX, 0, sizeof(matchX));
23     memset(matchY, 0, sizeof(matchY));
24     for(int i = 1; i <= R; ++i){
25         memset(visitX, false, sizeof(visitX));
26         memset(visitY, false, sizeof(visitY));
27         sum += dfs(i);
28     }
29     return sum;
30 }

```

```
31 int main(){
32     while(cin >> R >> C >> N && R && C && N){
33         memset(arr, false, sizeof(arr));
34         memset(visitX, false, sizeof(visitX));
35         memset(visitY, false, sizeof(visitY));
36         int row, col;
37         for(int i = 0; i < N; ++i){
38             cin >> row >> col;
39             arr[row][col] = true;
40         }
41         int cnt = Match();
42         cout << cnt;
43         memset(visitX, 0, sizeof(visitX));
44         memset(visitY, 0, sizeof(visitY));
45         for(int i = 1; i <= R; ++i){
46             if(matchX[i] == 0) dfs(i);
47         }
48         for(int i = 1; i <= R; ++i)
49             if(!visitX[i]) cout << " r" << i;
50         for(int i = 1; i <= C; ++i)
51             if(visitY[i]) cout << " c" << i;
52         cout << endl;
53     }
```