

Contents

| | | |
|------|-----------------------------|----|
| 1 | Sync | 1 |
| 1.1 | Sync | 1 |
| 2 | Data Structure | 1 |
| 2.1 | Binary Search | 1 |
| 2.2 | BIT | 1 |
| 2.3 | BWT | 1 |
| 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 |
| 5 | Enumerate | 2 |
| 5.1 | Halfcut Enumerate | 2 |
| 6 | Graph | 3 |
| 6.1 | SPFA | 3 |
| 6.2 | Dijkstra | 3 |
| 6.3 | Floyd Warshall | 3 |
| 6.4 | Disjoint set Kruskal | 3 |
| 6.5 | Bipartite | 4 |
| 6.6 | Hungarian algorithm | 4 |
| 6.7 | LCA | 4 |
| 6.8 | Trie | 5 |
| 7 | Math | 5 |
| 7.1 | Hash | 5 |
| 8 | Other | 5 |
| 8.1 | Ants Colony | 5 |
| 8.2 | Binary codes | 6 |
| 8.3 | Fire Fire Fire | 6 |
| 8.4 | Disk Tree | 6 |
| 8.5 | Stammering Aliens | 6 |
| 9 | DP | 7 |
| 9.1 | Crested Ibis vs Monster | 7 |
| 9.2 | dpd Knapsack 1 | 7 |
| 9.3 | Homer Simpson | 7 |
| 9.4 | Let Me Count The Ways | 8 |
| 9.5 | Luggage | 8 |
| 9.6 | Partitioning by Palindromes | 8 |
| 9.7 | SuperSale | 9 |
| 9.8 | Walking on the Safe Side | 9 |
| 9.9 | Cutting Sticks | 9 |
| 9.10 | Race to 1 | 9 |
| 9.11 | Apple | 10 |
| 9.12 | Stamps | 10 |
| 10 | LIS | 10 |
| 10.1 | Wavio Sequence | 10 |
| 10.2 | Robots II | 11 |
| 11 | Math | 11 |
| 11.1 | Big Mod | 11 |
| 11.2 | Bubble Sort Expect Value | 11 |
| 11.3 | Fraction Floor Sum | 11 |
| 11.4 | How Many Os | 12 |
| 11.5 | Number of Pairs | 12 |
| 11.6 | ORXOR | 12 |
| 11.7 | X drawing | 12 |
| 12 | Binary Search | 12 |
| 12.1 | Fill the Containers | 12 |
| 12.2 | Where is the marble | 13 |
| 13 | Segment Tree | 13 |
| 13.1 | Frequent values | 13 |
| 14 | Bipartite Graph | 14 |
| 14.1 | Claw Decomposition | 14 |
| 14.2 | Guardian of Decency | 14 |
| 14.3 | Taxi Cab Scheme | 14 |
| 15 | Function | 15 |
| 15.1 | CHAR | 15 |
| 15.2 | string | 15 |
| 15.3 | setprecision | 15 |
| 15.4 | GCD LCM | 15 |
| 15.5 | reverse | 15 |
| 15.6 | sort | 16 |
| 15.7 | map | 16 |
| 15.8 | set | 16 |

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    }
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)
16          dp[j][i] = dp[dp[j][i-1]][i-1];
17  for(int i = 0; i < Q; ++i){
18      cin >> a >> b;
19      a--; // 要減減是因為arr的index從0開始但題目從1開始
20      b--;
21      if(a > b) swap(a, b);
22      int ans = 0;
23      for(int i = LOG - 1; i >= 0; --i){ // 從後往前推
24          if(dp[a][i] < b){
25              ans += (1 << i);
26              a = dp[a][i];
27          }
28      }
29      cout << ans + 1 << endl;
30  }

```

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
10     for(int i = 0; i < a.size(); ++i)
11         maxi = max(maxi, len[i]);
12     return maxi;
13 }

```

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

```

```

21     long long int y = *--it);
22     if(x + y <= T) ans = max(ans, x + y);
23 }
24 cout << ans << endl;
25 }

```

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 Item{
3     int u, dis;
4     // 取路徑最短
5     bool operator < (const Item &other) const{
6         return dis > other.dis;
7     }
8 };
9 int dis[maxn];
10 vector<Edge> G[maxn];
11 void dijkstra(int s){
12     for(int i = 0; i <= n; i++){
13         dis[i] = inf;
14     }
15     dis[s] = 0;
16     priority_queue<Item> pq;
17     pq.push({s, 0});
18     while(!pq.empty()){
19         // 取路徑最短的點
20         Item now = pq.top();
21         pq.pop();
22         if(now.dis > dis[now.u]){
23             continue;
24         }
25         // 鬆弛更新，把與 now.u 相連的點都跑一遍
26         for(Edge e : G[now.u]){
27             if(dis[e.v] > now.dis + e.w){
28                 dis[e.v] = now.dis + e.w;

```

```

29         pq.push({e.v, dis[e.v]});
30     }
31 }
32 }
33 }

```

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, w;
3     // 用權重排序 由大到小
4     bool operator < (const Edge &other) const{
5         return w > other.w;
6     }
7 }edge[maxn];
8 // disjoint set
9 int find(int x){
10    if(parent[x] < 0){
11        return x;
12    }
13    else{
14        return parent[x] = find(parent[x]);
15    }
16 }
17 void unite(int a, int b){
18     a = find(a);
19     b = find(b);
20
21     if(a != b){
22         if(parent[a] < parent[b]){
23             parent[a] += parent[b];
24             parent[b] = a;
25         }
26         else{
27             parent[b] += parent[a];
28             parent[a] = b;
29         }
30     }
31 }
32 void kruskal(){
33     memset(parent, -1, sizeof(parent));
34     sort(edge, edge + m);
35     int i, j;
36     for(i = 0, j = 0; i < n - 1 && j < m; i++){
37         // 如果 u 和 v 的祖先相同，則 j++
38         // (祖先相同代表會產生環 所以不要)
39         while(find(edge[j].u) == find(edge[j].v)) j++;
40         // 若部會產生環 則讓兩點之間產生橋
41         // (連接兩顆子生成樹)
42         unite(edge[j].u, edge[j].v);
43         j++;
44     }

```

43 }

6.5 Bipatirate

```

1  /* 二分圖 */
2  const int maxn = 300 + 5;
3  int n, color[maxn];
4  vector<vector<int>> v(maxn);
5  bool dfs(int s){
6      for(auto it : v[s]){
7          if(color[it] == -1){
8              color[it] = 3 - color[s];
9              if(!dfs(it)){
10                 return false;
11             }
12         }
13         if(color[s] == color[it]){
14             return false;
15         }
16     }
17     return true;
18 }
19 void isBipatirate(){
20     bool flag = true;
21     for(int i = 1; i <= n; ++i){
22         if(color[i] == -1){
23             color[i] = 1;
24             flag &= dfs(i);
25         }
26     }
27     if(flag){
28         cout << "YES" << endl;
29     }
30     else{
31         cout << "NO" << endl;
32     }
33 }
34 int main(){
35     while(cin >> n && n){
36         for(int i = 1; i <= n; ++i) v[i].clear();
37         memset(color, -1, sizeof(color));
38         int a, b;
39         while(cin >> a >> b && (a || b)){
40             v[a].emplace_back(b);
41             v[b].emplace_back(a);
42         }
43         isBipatirate();
44     }
45 }

```

6.6 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 }

```

```

20 bool dfs(int s){
21     for(int i = 0; i < G[s].size(); ++i){
22         int v = G[s][i];
23         if(visited[v]) continue;
24         visited[v] = true;
25         if(match[v] == -1 || dfs(match[v])){
26             match[v] = s;
27             return true;
28         }
29     }
30     return false;
31 }
32 int Hungarian(){
33     int cnt = 0;
34     memset(match, -1, sizeof(match));
35     for(int i = 0; i < bn; ++i){
36         memset(visited, false, sizeof(visited));
37         if(dfs(i)) cnt++;
38     }
39     return cnt;
40 }
41 int main(){
42     cin >> t;
43     while(t--){
44         cin >> N;
45         bn = 0, gn = 0;
46         for(int i = 0; i <= N; ++i) G[i].clear();
47         int h;
48         string sex, music, sport;
49         for(int i = 0; i < N; ++i){
50             cin >> h >> sex >> music >> sport;
51             if(sex == "M") lef[bn++] = People(h,
52                 music, sport);
53             else rig[gn++] = People(h, music, sport);
54         }
55         for(int i = 0; i < bn; ++i){
56             for(int j = 0; j < gn; ++j)
57                 if(check(lef[i], rig[j]))
58                     G[i].emplace_back(j);
59         }
60         cout << N - Hungarian() << endl;
61     }
62 }

```

6.7 LCA

```

1  /*最低共同祖先*/
2  // 此 node 下有機顆 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] =
21                 parent[parent[j][i-1]][i-1];
22         }
23     }
24     // 求兩點的LCA (利用倍增法)
25     int LCA(int a, int b){
26         if(depth[b] < depth[a]) swap(a, b);
27         if(depth[a] != depth[b]){
28             int dif = depth[b] - depth[a];
29             for(int i = 0; i < 20; i++){
30                 if(dif & 1) b = parent[b][i];
31             }
32         }
33         while(a != b){
34             if(a > b) b = parent[b][1];
35             else a = parent[a][1];
36         }
37         return a;
38     }
39 }

```

```

29         dif >= 1;
30     }
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 }

```

6.8 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 = 40000+5;
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 }

```

8 Other

8.1 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     // 求兩點的LCA (利用倍增法)
34     int LCA(int a, int b){
35         if (depth[b] < depth[a]) swap(a, b);
36         if (depth[a] != depth[b]){
37             int dif = depth[b] - depth[a];
38             for (int i = 0; i < 20; i++){
39                 if (dif & 1) b = parent[b][i];
40                 dif >>= 1;
41             }
42         }
43         if (a == b) return a;
44         for (int i = 19; i >= 0; i--){
45             if (parent[a][i] != parent[b][i]){
46                 a = parent[a][i];
47                 b = parent[b][i];
48             }
49         }
50         return parent[a][0];
51     }
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         find_parent();
78         dfs(0, 0);
79         distance();
80         int s; cin >> s;
81         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]
            * 2);
84     }
85     cout << endl;
86 }
87 }

```

8.2 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         }
24         zero = 0;
25         BWT();
26     }
27 }

```

8.3 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  const int maxn = 1000+5;
4  int cnt = 0;
5  vector<int> G[maxn];
6  bool exi[maxn], visited[maxn];
7  void dfs(int node, int parent){
8      if(G[node].size() == 1 && G[node][0] == parent)
9          return;
10     for(int i = 0; i < G[node].size(); ++i){
11         int now = G[node][i];
12         if(visited[now]) continue;
13         visited[now] = true;
14         dfs(G[node][i], node);
15     }
16     bool flag = false;
17     for(int j = 0; j < G[node].size(); ++j){
18         if(exi[G[node][j]] != true && G[node][j] !=
19             parent){
20             flag = true;
21             break;
22         }
23     }
24     if(flag && exi[node] != true){
25         exi[node] = true;
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     }
35 }

```

```

32     memset(exi, false, sizeof(exi));
33     memset(visited, false, sizeof(visited));
34     for(int i = 1; i <= n; ++i){
35         int siz; cin >> siz;
36         for(int j = 0; j < siz; ++j){
37             int num; cin >> num;
38             G[i].emplace_back(num);
39         }
40     }
41     cnt = 0;
42     dfs(1, 1);
43     if(n == 1) cnt++;
44     cout << cnt << endl;
45 }
46 }

```

8.4 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     }
48 }

```

8.5 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;

```

9 DP

9.1 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         memset(dp, 0x3f3f3f3f, sizeof(dp));
19         dp[0][0] = 0;
20         for(int i = 1; i <= n; i++){
21             for(int j = 0; j <= h; j++){
22                 dp[i][j] = min(dp[i-1][j], dp[i][max(0, j
23                     - a[i])] + b[i]);
24             }
25         }
26     }

```

9.2 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
9 0 0 0 30 50 60 60 80 90 */
10 int main(){
11     int N, W;
12     cin >> N >> W;
13     int w[100000+5], v[100000+5];
14     for(int i = 0; i < N; i++){
15         cin >> w[i] >> v[i];
16     }
17     long long int dp[100000+5];
18     memset(dp, 0, sizeof(dp));
19     for(int i = 0; i < N; i++){
20         for(int j = W; j >= w[i]; j--){
21             dp[j] = max(dp[j], dp[j - w[i]] + v[i]);
22         }
23     }
24     cout << dp[W] << endl;

```

9.3 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){

```

```

23     for(int i = t; i >= 0; i--)
24         if(dp[i] != -1){
25             cout << dp[i] << " " << t - i <<
                endl;
26             break;
27         }
28     }
29     else cout << dp[t] << endl;
30 }
31 }

```

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

```

9.5 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  1
5  Note: dp[0] = true */
6  int main(){
7      int t;
8      cin >> t;
9      cin.ignore();
10     while(t--){
11         string str;
12         getline(cin, str);
13         vector<int> v;
14         stringstream ss;
15         int num, cnt = 0, sum = 0;
16         bool dp[4000+5];
17         memset(dp, false, sizeof(dp));
18         ss << str;
19         while(ss >> num){
20             cnt++;
21             sum += num;
22             v.emplace_back(num);
23         }
24         if(sum & 1){
25             cout << "NO" << endl;
26             continue;
27         }
28         dp[0] = true;
29         for(int i = 0; i < v.size(); i++){
30             for(int j = sum; j >= v[i]; j--){
31                 if(dp[j - v[i]])
32                     dp[j] = true;
33             }
34         }
35     }
36 }

```

```

32     cout << (dp[sum/2] ? "YES" : "NO") << endl;
33 }
34 }

```

9.6 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 }

```

9.7 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                 if(edge[j].p <= pw)
34                     dp[j+1] = max(dp[j], dp[j] + edge[j].w);
35             }
36             total += dp[g];
37             pw -= edge[g].p;
38         }
39         cout << total << endl;
40     }
41 }

```



```

28         for(int k = pw; k >= edge[j].w; k--)
29             dp[k] = max(dp[k], dp[k -
                    edge[j].w] + edge[j].p);
30         total += dp[pw];
31     }
32     cout << total << endl;
33 }
34 }

```

9.8 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 }

```

9.9 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         cout << "The minimum cutting is " << dp[0][n]
45         << "." << endl;
46     }
47 }

```

9.10 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;

```

```

38 Linear_Sieve();
39 cin >> t;
40 while(t--){
41     cin >> num;
42     cout << "Case " << ca++ << ": " << fixed <<
         setprecision(10) << dfs(num) << endl;
43 }
44 }

```

9.11 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 }

```

9.12 Stamps

```

1 /* DFS 處理 DP */
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;

```

```

32     }
33     // 更新現在連續最大值到多少
34     while(y[r+1] < 0x3f3f3f) r++;
35     // x 可貼面額種類多 1
36     dfs(i+1);
37     // 還原 r、y 值
38     r = r_before_this_layer;
39     for(int j = 0; j < maxn; ++j)
40         y[j] = y_before_this_layer[j];
41 }
42 }
43 int main(){
44     while(cin >> h >> k && h && k){
45         memset(x, 0, sizeof(x));
46         memset(y, 0x3f3f3f3f, sizeof(y));
47         x[0] = 1;
48         r = h;
49         // x[0] = 1, 1 張郵票可貼到的最大值
50         for(int i = 0; i <= r; ++i)
51             y[i] = i;
52         maxi = 0;
53         dfs(1);
54         for(int i = 0; i < k; ++i)
55             printf("%3d", ans[i]);
56         printf(" ->%3d\n", maxi);
57     }
58 }

```

10 LIS

10.1 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     cout << maxi * 2 - 1 << endl;
52 }
53 }

```

10.2 Robots II

```

1  /* LIS
2  No.:  2  4 11 13 25 28 41 42
3  LIS:  1  2  3  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 <<
        " " << G[n];
14 }
15 int main(){
16     int ca = 1;
17     while(cin >> r >> c && (r != -1) && (c != -1)){
18         memset(garbage, false, sizeof(garbage));
19         G.clear();
20         int x, y;
21         while(cin >> x >> y && x && y){
22             garbage[x][y] = true;
23         }
24         // 紀錄有垃圾的點的編號
25         for(int i = 1; i <= r; ++i){
26             for(int j = 1; j <= c; ++j){
27                 if(garbage[i][j]) G.emplace_back((i -
                    1) * c + j);
28             }
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) %
                    c)){
42                     // num 是經過的路徑數量。path
43                     // 是從誰來
44                     if(LIS[i] == LIS[j] + 1){
45                         num[i] += num[j];
46                     }
47                     else if(LIS[i] < LIS[j] + 1){
48                         LIS[i] = LIS[j] + 1;
49                         num[i] = num[j];
50                         path[i] = j;
51                     }
52                 }
53             }
54             G.pop_back();
55             // 要把假設還回去
56             if(!garbage[r][c]) LIS[G.size() - 1]--;

```

```

56     cout << "CASE#" << ca++ << ": " <<
        LIS[G.size() - 1] << " " << num[G.size()
            - 1];
57     show_path(G.size() - 1);
58     cout << endl;
59 }
60 }

```

11 Math

11.1 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

```

11.2 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 ) << "/2" << endl;
18     }
19     else{
20         cout << ( (n * (n - 1)) / 2 ) / 2 << endl;
21     }

```

11.3 Fraction Floor Sum

```

1  /* 數論
2  [N/i] == M
3  -> M <= N/i < M + 1
4  -> N/(M+1) < i <= 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 }

```

11.4 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 }

```

11.5 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 }

```

11.6 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     }
28     // 不知道為什麼只有 2147483647 給過
29     int mini = 2147483647;
30     // 1 << n = n * 2
31     for(int i = 0; i < (1 << n); i++){
32         int XOR = 0, OR = 0;
33         for(int j = 1; j <= n; j++){
34             OR |= num[j];
35             if((i & (1 << j))){
36                 XOR ^= OR;
37             }
38             XOR |= OR;
39             mini = min(mini, XOR);
40         }
41     }
42     cout << mini << endl;
43 }

```

11.7 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             cout << endl;
13         }
14     }

```

12 Binary Search

12.1 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;

```

```

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

```

12.2 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 }

```

13 Segement Tree

13.1 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，就不會切到區塊
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答案
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             }
79         }
80     }

```

```

70         start = num[i].lef = i;
71     }
72     else num[i].lef = start;
73 }
74 // 最後一段 [start, n]
75 for(int j = start; j <= n; ++j){
76     num[j].rig = n;
77     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 }

```

14 Bipartite Graph

14.1 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
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 }

```

14.2 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 ==
21     girl.music && boy.sport != girl.sport) return
22     true;
23     return false;
24 }
25 bool dfs(int s){
26     for(int i = 0; i < G[s].size(); ++i){
27         int v = G[s][i];
28         if(visited[v]) continue;
29         visited[v] = true;
30         // 如果這個女生還沒被配對過，直接匹配
31         // 如果已經被配對，則根據這個女生所配對的對象
32         dfs 重新匹配所有人的對象
33         if(match[v] == -1 || dfs(match[v])){
34             match[v] = s;
35             return true;
36         }
37     }
38     return false;
39 }
40 int Hungarian(){
41     int cnt = 0;
42     memset(match, -1, sizeof(match));
43     for(int i = 0; i < bn; ++i){
44         memset(visited, false, sizeof(visited));
45         if(dfs(i)) cnt++;
46     }
47     return cnt;
48 }
49 int main(){
50     int t;
51     cin >> t;
52     while(t--){
53         int N;
54         cin >> N;
55         bn = 0, gn = 0;
56         for(int i = 0; i <= N; ++i) G[i].clear();
57         int h;
58         string sex, music, sport;
59         for(int i = 0; i < N; ++i){
60             cin >> h >> sex >> music >> sport;
61             if(sex == "M")
62                 lef[bn++] = People(h, music, sport);
63             else
64                 rig[gn++] = People(h, music, sport);
65         }
66         for(int i = 0; i < bn; ++i)
67             for(int j = 0; j < gn; ++j)
68                 if(check(lef[i], rig[j]))
69                     G[i].emplace_back(j);
70         cout << N - Hungarian() << endl;
71     }
72 }

```

14.3 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) +
16             abs(boy.y2 - boy.y1) + abs(boy.x2 - girl.x1)
17             + abs(boy.y2 - girl.y1);
18     if(tmp < girl.s) return true;
19     return false;
20 }
21 bool dfs(int s){
22     for(int i = 0; i < G[s].size(); ++i){
23         int v = G[s][i];
24         if(visited[v]) continue;
25         visited[v] = true;
26         if(match[v] == -1 || dfs(match[v])){
27             match[v] = s;
28             return true;
29         }
30     }
31     return false;
32 }
33 int Hungarian(){
34     int cnt = 0;
35     memset(match, -1, sizeof(match));
36     for(int i = 0; i < n; ++i){
37         memset(visited, false, sizeof(visited));
38         if(dfs(i)) cnt++;
39     }
40     return cnt;
41 }
42 int main(){
43     int t;
44     scanf("%d", &t);
45     while(t--){
46         scanf("%d", &n);
47         for(int i = 0; i < n; ++i) G[i].clear();
48         for(int i = 0; i < n; ++i){
49             int h, m;
50             scanf("%d:%d", &h, &m);
51             p[i].s = h * 60 + m;
52             scanf("%d%d%d%d", &p[i].x1, &p[i].y1,
53                     &p[i].x2, &p[i].y2);
54         }
55         sort(p, p + n);
56         for(int i = 0; i < n; ++i)
57             for(int j = i + 1; j < n; ++j)
58                 if(check(p[i], p[j]))
59                     G[i].push_back(j);
60         printf("%d\n", n - Hungarian());
61     }
62 }

```

15 Function

15.1 CHAR

```

1 isdigit()
2 isalnum() // 判斷字母 // 數字
3 isalpha()
4 islower()
5 isupper()
6 isblank() // 判斷 即 space 和 \t
7 toupper()

```

```
8 tolower()
```

15.2 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    }
26 }

```

15.3 setprecision

```

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

```

15.4 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 ) !=
12         0 );

```

15.5 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

```

15.6 sort

```

1 priority_queue<int, vector<int>, less<int>> // 大到小
2 priority_queue<int, vector<int>, greater<int>> //
   小到大
3
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

```

```

20 auto it = st.find(x); // binary search, O(log(N))
21 auto it = st.lower_bound(x); // binary search,
   O(log(N))
22 auto it = st.upper_bound(x); // binary search,
   O(log(N))
23
24 st.clear();
25 }

```

15.7 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 }

```

15.8 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);

```