

Contents

1	Data Structure	1
1.1	Binary Search	1
1.2	BIT	1
1.3	Trie	1
1.4	BWT	2
2	Divide and Conquer	2
2.1	count inversions	2
3	DP	2
3.1	Doubling	2
3.2	Josephus	2
3.3	LCS	2
3.4	LIS	2
4	Enumerate	2
4.1	Halfcut Enumerate	2
5	Graph	3
5.1	SPFA	3
5.2	Dijkstra	3
5.3	Floyd Warshall	3
5.4	Disjoint set Kruskal	3
5.5	KM	4
5.6	Dinic	4
5.7	Bipartite	5
5.8	Hungarian algorithm	5
5.9	LCA	5
6	Other	6
6.1	Ants Colony	6
6.2	Binary codes	6
7	DP	7
7.1	Crested Ibis vs Monster	7
7.2	dpd Knapsack 1	7
7.3	Homer Simpson	7
7.4	Let Me Count The Ways	7
7.5	Luggage	7
7.6	Partitioning by Palindromes	8
7.7	SuperSale	8
7.8	Walking on the Safe Side	8
7.9	Cutting Sticks	9
7.10	Race to 1	9
8	Math	9
8.1	Big Mod	9
8.2	Bubble Sort Expect Value	9
8.3	Fraction Floor Sum	10
8.4	How Many Os	10
8.5	Number of Pairs	10
8.6	ORXOR	10
8.7	X drawing	11
9	Binary Search	11
9.1	Fill the Containers	11
9.2	Where is the marble	11
10	Segment Tree	11
10.1	Frequent values	11
11	Bipartite Graph	12
11.1	Claw Decomposition	12
11.2	Guardian of Decency	12
12	Function	13
12.1	strstr	13
12.2	substr	13
12.3	map set	13
12.4	vector	13
12.5	setprecision	13
12.6	GCD LCM	13
12.7	reverse	13
12.8	CHAR	13
12.9	sort	14
12.10	struct	14
12.11	deque	14
12.12	python template	14

1 Data Structure

1.1 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 }

```

1.2 BIT

```

1 #define lowbit(k) (k & -k)
2 void add(vector<int> &tr, int id, int val) {
3     for (; id <= n; id += lowbit(id)) {
4         tr[id] += val;
5     }
6 }
7 int sum(vector<int> &tr, int id) {
8     int ret = 0;
9     for (; id >= 1; id -= lowbit(id)) {
10        ret += tr[id];
11    }
12    return ret;
13 }

```

1.3 Trie

```

1 const int MAXL = ; // 自己填
2 const int MAXC = ;
3 struct Trie {
4     int nex[MAXL][MAXC];
5     int len[MAXL];
6     int sz;
7     void init() {
8         memset(nex, 0, sizeof(nex));
9         memset(len, 0, sizeof(len));
10        sz = 0;
11    }
12    void insert(const string &str) {
13        int p = 0;
14        for (char c : str) {
15            int id = c - 'a';
16            if (!nex[p][id]) {
17                nex[p][id] = ++sz;
18            }
19            p = nex[p][id];
20        }
21        len[p] = str.length();
22    }
23    vector<int> find(const string &str, int i) {
24        int p = 0;
25        vector<int> ans;
26        for (; i < str.length(); i++) {
27            int id = str[i] - 'a';
28            if (!nex[p][id]) {
29                return ans;
30            }
31            p = nex[p][id];
32            if (len[p]) {
33                ans.pb(len[p]);
34            }
35        }
36    }
37 }

```

```

36     return ans;
37 }
38 };

```

1.4 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     }

```

2 Divide and Conquer

2.1 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         for(int k = lef; k < rig; ++k) arr[k] = buf[k];
21         return ans;
22     }

```

3 DP

3.1 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     }

```

3.2 Josephus

```

1  int josephus (int n, int k) {
2      // 有 n 個人圍成一圈，每 k 個一次
3      return n > 1 ? (josephus(n-1, k) + k) % n : 0;
4  }
5  // 回傳最後一人的編號，0 index

```

3.3 LCS

```

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

```

3.4 LIS

```

1  int LIS(vector<int> &a) { // Longest Increasing
2      Subsequence
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 Enumerate

4.1 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      }
7  }

```

```

5     return;
6 }
7 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  $S_x$  的數字  $S_{xi}$ , 找出  $S_y$ 
19     // 集合內小於等於  $T - S_{xi}$  中最大的數  $S_{yj}$ 
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;
26 }

```

```

11     for(int i = 0; i <= n; i++){
12         dis[i] = inf;
13     }
14     dis[s] = 0;
15     priority_queue<Item> pq;
16     pq.push({s, 0});
17     while(!pq.empty()){
18         // 取路徑最短的點
19         Item now = pq.top();
20         pq.pop();
21         if(now.dis > dis[now.u]){
22             continue;
23         }
24         // 鬆弛更新, 把與 now.u 相連的點都跑一遍
25         for(Edge e : G[now.u]){
26             if(dis[e.v] > now.dis + e.w){
27                 dis[e.v] = now.dis + e.w;
28                 pq.push({e.v, dis[e.v]});
29             }
30         }
31     }
32 }

```

5 Graph

5.1 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 }

```

5.2 Dijkstra

```

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

```

5.3 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 }

```

5.4 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         // 若都會產生環 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     }
45 }

```

5.5 KM

```

1  const int X = 50; // x的點數, 等於y的點數
2  const int Y = 50; // y的點數
3  int adj[X][Y]; // 精簡過的adjacency matrix
4  int lx[X], ly[Y]; // vertex labeling
5  int mx[X], my[Y]; //
6  // x各點的配對對象、y各點的配對對象
7  int q[X], *qf, *qb; // BFS queue
8  int p[X]; // BFS
9  // parent, 交錯樹之偶點, 指向上一個偶點
10 bool vx[X], vy[Y]; // 記錄是否在交錯樹上
11 int dy[Y], pdy[Y]; // 表格
12
13 void relax(int x){ // relaxation
14     for (int y=0; y<Y; ++y)
15         if (adj[x][y] != 1e9)
16             if (lx[x] + ly[y] - adj[x][y] < dy[y]){
17                 dy[y] = lx[x] + ly[y] - adj[x][y];
18                 pdy[y] = x; //
19                 // 記錄好是從哪個樹葉連出去的
20             }
21 }
22
23 void reweight(){ // 調整權重、調整表格
24     int d = 1e9;
25     for (int y=0; y<Y; ++y) if (!vy[y]) d = min(d, dy[y]);
26     for (int x=0; x<X; ++x) if (vx[x]) lx[x] -= d;
27     for (int y=0; y<Y; ++y) if (vy[y]) ly[y] += d;
28     for (int y=0; y<Y; ++y) if (!vy[y]) dy[y] -= d;
29 }
30
31 void augment(int x, int y){ // 擴充路徑
32     for (int ty; x != -1; x = p[x], y = ty){
33         ty = mx[x]; my[y] = x; mx[x] = y;
34     }
35 }
36
37 bool branch1(){ // 延展交錯樹: 使用既有的等邊
38     while (qf < qb)
39         for (int x=*qf++, y=0; y<Y; ++y)
40             if (!vy[y] && lx[x] + ly[y] == adj[x][y]){
41                 vy[y] = true;
42                 if (my[y] == -1){
43                     augment(x, y);
44                     return true;
45                 }
46                 int z = my[y];
47                 *qb++ = z; p[z] = x; vx[z] = true;
48                 relax(z);
49             }
50     return false;
51 }
52
53 bool branch2(){ // 延展交錯樹: 使用新添的等邊
54     for (int y=0; y<Y; ++y){
55         if (!vy[y] && dy[y] == 0){
56             vy[y] = true;

```

```

49         if (my[y] == -1){
50             augment(pdy[y], y);
51             return true;
52         }
53         int z = my[y];
54         *qb++ = z; p[z] = pdy[y]; vx[z] = true;
55         relax(z);
56     }
57     return false;
58 }
59 int Hungarian(){
60     // 初始化vertex labeling
61     // memset(lx, 0, sizeof(lx)); // 任意值皆可
62     memset(ly, 0, sizeof(ly));
63     for (int x=0; x<X; ++x)
64         for (int y=0; y<Y; ++y)
65             lx[x] = max(lx[x], adj[x][y]);
66
67     // x側每一個點, 分別建立等邊交錯樹。
68     memset(mx, -1, sizeof(mx));
69     memset(my, -1, sizeof(my));
70     for (int x=0; x<X; ++x){
71         memset(vx, false, sizeof(vx));
72         memset(vy, false, sizeof(vy));
73         memset(dy, 0x7f, sizeof(dy));
74         qf = qb = q;
75         *qb++ = x; p[x] = -1; vx[x] = true; relax(x);
76         while (true){
77             if (branch1()) break;
78             reweight();
79             if (branch2()) break;
80         }
81     }
82     // 計算最大權完美匹配的權重
83     int weight = 0;
84     for (int x=0; x<X; ++x)
85         weight += adj[x][mx[x]];
86     return weight;
87 }

```

5.6 Dinic

```

1  // Maximum Flow
2  const int V = 100, E = 1000;
3  int adj[V]; // adjacency lists, 初始化為-1。
4  struct Element {int b, r, next;} e[E*2];
5  int en = 0;
6  void addedge(int a, int b, int c){
7      e[en] = (Element){b, c, adj[a]}; adj[a] = en++;
8      e[en] = (Element){a, 0, adj[b]}; adj[b] = en++;
9  }
10 int d[V]; // 最短距離
11 bool visit[V]; // BFS/DFS visit record
12 int q[V]; // queue
13 int BFS(int s, int t){ // 計算最短路徑, 求出容許圖
14     memset(d, 0x7f, sizeof(d));
15     memset(visit, false, sizeof(visit));
16     int qn = 0;
17     d[s] = 0;
18     visit[s] = true;
19     q[qn++] = s;
20
21     for (int qf=0; qf<qn; ++qf){
22         int a = q[qf];
23         for (int i = adj[a]; i != -1; i = e[i].next){
24             int b = e[i].b;
25             if (e[i].r > 0 && !visit[b]){
26                 d[b] = d[a] + 1;
27                 visit[b] = true;
28                 q[qn++] = b;
29                 if (b == t) return d[t];
30             }
31         }
32     }
33 }

```

```

33     return V;
34 }
35 int DFS(int a, int df, int s, int t){ //
    求出一條最短擴充路徑，並擴充流量
36     if (a == t) return df;
37     if (visit[a]) return 0;
38     visit[a] = true;
39     for (int i = adj[a]; i != -1; i = e[i].next){
40         int b = e[i].b;
41         if (e[i].r > 0 && d[a] + 1 == d[b]){
42             int f = DFS(b, min(df, e[i].r), s, t);
43             if (f){
44                 e[i].r -= f;
45                 e[i^1].r += f;
46                 return f;
47             }
48         }
49     }
50     return 0;
51 }
52 int dinitz(int s, int t){
53     int flow = 0;
54     while (BFS(s, t) < V)
55         while (true){
56             memset(visit, false, sizeof(visit));
57             int f = DFS(s, 1e9, s, t);
58             if (!f) break;
59             flow += f;
60         }
61     return flow;
62 }

```

5.7 Bipatirate

```

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

```

```

42         isBipatirate();
43     }
44 }

```

5.8 Hungarian algorithm

```

1  const int maxn = 500+5;
2  int t, N, bn, gn, match[maxn];
3  bool visited[maxn];
4  vector<vector<int>> G(maxn);
5  struct People{
6      int h;
7      string music, sport;
8      People(){}
9      People(int h, string music, string sport){
10         this->h = h;
11         this->music = music;
12         this->sport = sport;
13     }
14 }lef[maxn], rig[maxn];
15 bool check(People boy, People girl){
16     if(abs(boy.h - girl.h) <= 40 && boy.music ==
        girl.music && boy.sport != girl.sport) 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 < bn; ++i){
35         memset(visited, false, sizeof(visited));
36         if(dfs(i)) cnt++;
37     }
38     return cnt;
39 }
40 int main(){
41     cin >> t;
42     while(t--){
43         cin >> N;
44         bn = 0, gn = 0;
45         for(int i = 0; i <= N; ++i) G[i].clear();
46         int h;
47         string sex, music, sport;
48         for(int i = 0; i < N; ++i){
49             cin >> h >> sex >> music >> sport;
50             if(sex == "M") lef[bn++] = People(h,
                music, sport);
51             else rig[gn++] = People(h, music, sport);
52         }
53         for(int i = 0; i < bn; ++i){
54             for(int j = 0; j < gn; ++j)
55                 if(check(lef[i], rig[j]))
56                     G[i].emplace_back(j);
57         }
58         cout << N - Hungarian() << endl;
59     }

```

5.9 LCA

```
1  /*最低共同祖先*/

```

```

2 // 此 node 下有機類 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     }
33     if (a == b) return a;
34     for (int i = 19; i >= 0; i--){
35         if (parent[a][i] != parent[b][i]){
36             a = parent[a][i];
37             b = parent[b][i];
38         }
39     }
40     return parent[a][0];
41 }

```

6 Other

6.1 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 下有機類 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] =
                parent[parent[j][i-1]][i-1];
30         }
31     }
32 // 求兩點的 LCA (利用倍增法)
33 int LCA(int a, int b){
34     if (depth[b] < depth[a]) swap(a, b);
35     if (depth[a] != depth[b]){
36         int dif = depth[b] - depth[a];
37         for (int i = 0; i < 20; i++){
38             if (dif & 1) b = parent[b][i];
39             dif >>= 1;
40         }
41     }
42     if (a == b) return a;
43     for (int i = 19; i >= 0; i--){
44         if (parent[a][i] != parent[b][i]){
45             a = parent[a][i];
46             b = parent[b][i];
47         }
48     }
49     return parent[a][0];
50 }
51 long long int dist[maxn];
52 // 從 0 開始到每個點的距離
53 void distance(){
54     for (int u = 0; u < N; ++u){
55         for(int i = 0; i < G[u].size(); ++i){
56             dist[G[u][i].v] = dist[u] + G[u][i].w;
57         }
58     }
59 int main(){
60     while(cin >> N && N){
61         memset(dist, 0, sizeof(dist));
62         memset(parent, 0, sizeof(parent));
63         memset(depth, 0, sizeof(depth));
64         memset(siz, 0, sizeof(siz));
65         for(int i = 0; i <= N; ++i){
66             G[i].clear();
67         }
68         for(int i = 1; i < N; ++i){
69             int u, w;
70             cin >> u >> w;
71             G[u].push_back({i, w});
72             parent[i][0] = u;
73         }
74         find_parent();
75         dfs(0, 0);
76         distance();
77         int s; cin >> s;
78         bool space = false;
79         for(int i = 0; i < s; ++i){
80             int a, b;
81             cin >> a >> b;
82             int lca = LCA(a, b);
83             if(space) cout << " ";
84             space = true;
85             cout << (dist[a] + dist[b]) - (dist[lca]
86                 * 2);
87         }
88         cout << endl;
89     }
90 }

```

6.2 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        }
12        int ptr = mini[0];
13        for(int i = 0; i < n; ++i){
14            cout << back[ptr] << " ";
15        }
16    }
17 }

```

```

12     ptr = mini[ptr];
13 }
14     cout << endl;
15 }
16 int main(){
17     cin >> n;
18     for(int i = 0; i < n; ++i){
19         cin >> back[i];
20         zero = 0;
21         BWT();
22     }

```

7 DP

7.1 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     }

```

7.2 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;
25 }

```

7.3 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             else cout << dp[t] << endl;
39         }
40     }
41 }

```

7.4 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     while(cin >> n){
18         if(dp[n] == 1)
19             cout << "There is only " << dp[n] << "
20                 way to produce " << n << " cents
21                 change." << endl;
22         else
23             cout << "There are " << dp[n] << " ways
24                 to produce " << n << " cents change."
25                 << endl;
26     }
27 }

```

7.5 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

```



```

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 }

```

7.6 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 }

```

7.7 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 - edge[j].w] + edge[j].p);
35                 }
36                 total += dp[pw];
37             }
38             cout << total << endl;
39         }
40     }
41 }

```

7.8 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 }

```

7.9 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] = l;
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 }

```

7.10 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     }
47 }

```

8 Math

8.1 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

```

8.2 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 *
4          1/6 + 6 * 1/6
5  = (1 + 2 + 3 + 4 + 5 + 6)/6 = 3.5
6  bubble sort 每兩兩之間交換機率是 1/2
7  總共會做 C(n, 2) 次
8  E(x) = C(n, 2) * 1/2 = (n * (n - 1))/2 * 1/2 */
9  int t, ca = 1;
10  cin >> t;
11  while(t--){
12      long long int n;

```

```

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

```

8.3 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 }

```

8.4 How Many Os

```

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 }

```

8.5 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 }

```

8.6 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 int main(){
21     int n; cin >> n;
22     int num[20+7];
23     memset(num, 0, sizeof(num));
24     for(int i = 1; i <= n; i++){
25         cin >> num[i];
26         // 不知道為甚麼只有 2147483647 給過
27         int mini = 2147483647;
28         // 1 << n = n * 2
29         for(int i = 0; i < (1 << n); i++){
30             int XOR = 0, OR = 0;
31             for(int j = 1; j <= n; j++){
32                 OR |= num[j];
33                 if((i & (1 << j))){
34                     XOR ^= OR;
35                     OR = 0;
36                 }
37             }
38             XOR ^= OR;
39             mini = min(mini, XOR);
40         }
41     }
42 }

```

```

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

```

8.7 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     }

```

9 Binary Search

9.1 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         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         }

```

9.2 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 }

```

10 Segement Tree

10.1 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都在同個區塊上，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[Lson(x)].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

```

```

41     lef = num[lef].rig + 1;
42 }
43 // 查詢的右區間邊界切到區塊，且此區間有數個區塊
44 if(rig < num[rig].rig){
45     // 計算切到的區間大小，並找出最大
46     ans = max(ans, rig - num[rig].lef + 1);
47     // 更新右邊界
48     rig = num[rig].lef - 1;
49 }
50 //
51 // 如果左邊界大於右邊界，表示不需要再進行查詢直接回傳
52 if(lef > rig) return ans;
53 if(tree[x].lef >= lef && tree[x].rig <= rig)
54     return tree[x].value;
55 int mid = (tree[x].lef + tree[x].rig) >> 1;
56 if(lef <= mid) ans = max(ans, query(lef, rig,
57     Lson(x)));
58 if(mid < rig) ans = max(ans, query(lef, rig,
59     Rson(x)));
60 return ans;
61 }
62 int main(){
63     int n, q;
64     while(cin >> n && n){
65         cin >> q;
66         int start = 1;
67         for(int i = 1; i <= n; ++i){
68             cin >> num[i].value;
69             if(num[i].value != num[i-1].value){
70                 for(int j = start; j < i; ++j){
71                     num[j].rig = i - 1;
72                     num[j].fre = i - start;
73                 }
74                 start = num[i].lef = i;
75             }
76             else num[i].lef = start;
77         }
78         // 最後一段 [start, n]
79         for(int j = start; j <= n; ++j){
80             num[j].rig = n;
81             num[j].fre = n - start + 1;
82         }
83         build(1, n, 1);
84         int lef, rig;
85         for(int i = 0; i < q; ++i){
86             cin >> lef >> rig;
87             cout << query(lef, rig, 1) << endl;
88         }
89     }
90 }

```

11 Bipartite Graph

11.1 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 }

```

```

18     }
19     return true;
20 }
21 void isBipatirate(){
22     bool flag = true;
23     for(int i = 1; i <= n; ++i){
24         if(color[i] == -1){
25             // 如果還未填色過，就先填色成
26             // 1，並對與此點相連的點都 dfs 判定填色
27             color[i] = 1;
28             flag &= dfs(i);
29         }
30     }
31     if(flag) cout << "YES" << endl;
32     else cout << "NO" << endl;
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 }

```

11.2 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 }

```

```

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 }

```

12 Function

12.1 strstr

```

1 #include <stdio.h>
2 #include <string.h>
3
4 int main(){
5     char * c;
6     char str1[1005], str2[1005];
7     scanf("%s %s", str1, str2);
8     c = strstr(str1, str2);
9     if (c != NULL){
10         printf("Yes\n");
11     }
12     else printf("No\n");
13 }
14 // Input : Hello eLl
15 // Output : No

```

12.2 substr

```

1 int main(){
2     string str; //abcdef
3     cin >> str;
4     string tmp;
5     tmp = str.substr(0, 2); //ab
6     str = str.substr(2); //cdef
7     cout << tmp << " " << str;
8     return 0;
9 }

```

12.3 map set

```

1 .begin( ) // Return iterator to beginning
2 .end( ) // Return iterator to end
3 .empty( ) // 檢查是否為空
4 .size( ) // 回傳大小
5 mp.insert(pair<char,int>('a',100))
6 st.insert(100) // 插入key、value
7 .erase( ) // 刪掉指定key和他的value

```

```

8 .clear( ) // 清空整個 map
9 m.find( )
10 cout << "a => " << mymap.find('a')->second << endl;
11 // 找出 map 裡 key
12 // 有沒有在裡面，如果有的話會回傳元素所在的 iterator，否則傳 end()
13 s.count() // 返回某個值元素在 set 的個數
14 while( !mymap.empty()){
15     cout << mymap.begin()->first << " => " <<
16         mymap.begin()->second << endl;
17     mymap.erase(mymap.begin());
18 }
19 for (auto it = mymap.begin(); it != mymap.end(); ++it)
20     cout << it->first << " => " << it->second << endl;

```

12.4 vector

```

1 v.erase(v.begin() + 5) //拿掉第六個數
2 v.erase (v.begin(), v.begin() + 3); //拿掉前三個數

```

12.5 setprecision

```

1 // 將數字的小數部分設定為固定長度
2 cnt = 3.5555;
3 cout << fixed << setprecision(3) << cnt ;
4 // output : 3.555

```

12.6 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 );

```

12.7 reverse

```

1 int a[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
2 reverse(a, a+5) // 轉換0~5
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

```

12.8 CHAR

```

1 isdigit()
2 isalnum() //判斷字母 // 數字
3 isalpha()
4 islower()
5 isupper()
6 isblank() //判斷是否為空格，或者 tab 健制表符，即
7     space 和 \t
8 toupper()
9 tolower()

```

12.9 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

```

```

26         # e 次 x 幂 math.exp(x)
27     except EOFError:
28         pass

```

12.10 struct

```

1 struct area{
2     int a, b;
3     bool operator<(const area rhs) const{
4         return a > rhs.a || ( a == a && b > rhs.b);
5     }
6     bool operator!=(const area rhs) const{
7         return a != rhs.a || b != rhs.b;
8     }
9 };

```

12.11 deque

```

1 deque<int> que;
2 que.push_back(10);
3 que.push_front(20);
4 que.front()
5 que.back()
6 que.pop_front()
7 que.pop_back()
8 cout << "Element at position 2 : " << que.at(2) <<
   endl;

```

12.12 python template

```

1 import math
2 import operator
3
4 try:
5     while(1):
6         listx = []
7         listx.append("...")
8         list_s = sorted(listx) # 小到大
9         list_s = sorted(listx, reverse = True) #
           大到小
10        # max(listx)
11        # min(listx)
12        # sum(listx)
13        # len(listx)
14        dicty = {}
15        dicty[key] = "value"
16        dicty= sorted(dicty.items()) # by key
17        dicty= sorted(dicty.items(),
           key=operator.itemgetter(1)) # by value
18        # EOF寫法
19        # 階層 math.factorial(3) == 6
20        # 絕對值 math.fabs(x)
21        # 無條件進位 math.ceil(3.1) == 3
22        # 無條件捨去 math.floor(2.9) == 2
23        # C n 取 k math.comb(n, k)
24        # math.gcd
25        # math.lcm

```