

Contents

1 Sync	1	16 Dijkstra	18
1.1 Sync	1	16.1 Airport Express	18
2 Data Structure	1	16.2 Walk Through the Forest	19
2.1 Binary Search	1	17 Kruskal	19
2.2 BIT	1	17.1 Qin Shi Huang Road System	19
2.3 BWT	2	18 Bipartite Graph	20
3 Divide and Conquer	2	18.1 Claw Decomposition	20
3.1 count inversions	2	18.2 Guardian of Decency	20
4 DP	2	18.3 Taxi Cab Scheme	21
4.1 Doubling	2	18.4 SAM I AM	21
4.2 LCS	2	19 Function	22
4.3 LIS	2	19.1 CHAR	22
4.4 LIS 2	2	19.2 string	22
4.5 Minimum Edit Distance	2	19.3 setprecision	22
5 Enumerate	3	19.4 GCD LCM	22
5.1 Halfcut Enumerate	3	19.5 reverse	22
6 Graph	3	19.6 sort	22
6.1 SPFA	3	19.7 map	22
6.2 Dijkstra	3	19.8 set	22
6.3 Floyd Warshall	4	1 Sync	
6.4 Disjoint set Kruskal	4	1.1 Sync	
6.5 Disjoint set Kruskal 2	4		
6.6 Bipartite	4		
6.7 Hungarian algorithm	5		
6.8 LCA	5		
6.9 Trie	5		
7 Math	6		
7.1 Hash	6		
8 Other	6		
8.1 Ants Colony	6		
8.2 Binary codes	6		
8.3 Fire Fire Fire	7		
8.4 Disk Tree	7		
8.5 Stammering Aliens	7		
8.6 Fabled Rooks	8		
8.7 Rails	8		
8.8 String Distance and Transform Process	9		
9 Greedy	9		
9.1 Sticks	9		
10 DP	10		
10.1 Crested Ibis vs Monster	10		
10.2 dpd Knapsack 1	10		
10.3 Homer Simpson	10		
10.4 Let Me Count The Ways	10		
10.5 Luggage	10		
10.6 Partitioning by Palindromes	11		
10.7 SuperSale	11		
10.8 Walking on the Safe Side	11		
10.9 Cutting Sticks	12		
10.10 Race to 1	12		
10.11 Apple	12		
10.12 Stamps	12		
10.13 Evacuation Plan	13		
10.14 Ladies Choice	13		
11 LIS	14		
11.1 Wavio Sequence	14		
11.2 Robots II	14		
12 Math	15		
12.1 Big Mod	15		
12.2 Bubble Sort Expect Value	15		
12.3 Fraction Floor Sum	15		
12.4 How Many Os	15		
12.5 Number of Pairs	15		
12.6 ORXOR	16		
12.7 X drawing	16		
12.8 Playing With Stones	16		
12.9 And Then There Was One	16		
13 Binary Search	16		
13.1 Fill the Containers	16		
13.2 Where is the marble	17		
14 Graph	17		
14.1 Maximum sum on a torus	17		
15 Segment Tree	17		
15.1 Frequent values	17		

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--;

```

```

19     if(a > b) swap(a, b);
20     int ans = 0;
21     for(int i = LOG - 1; i >= 0; --i){ // 從後往前推
22         if(dp[a][i] < b){
23             ans += (1 << i);
24             a = dp[a][i];
25         }
26     }
27     cout << ans + 1 << endl;
28 }

```

4.2 LCS

```

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

```

4.3 LIS

```

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

```

4.4 LIS 2

```

1  int LIS(vector<int> &a){
2      int len[a.size()];
3      for(int i = 0; i < a.size(); ++i) len[i] = 1;
4      int maxi = -1;
5      for(int i = 0; i < a.size(); ++i)
6          for(int j = i + 1; j < a.size(); ++j)
7              if(a[i] <= a[j]) len[j] = max(len[j],
8                                              len[i] + 1);
9
10     for(int i = 0; i < a.size(); ++i)
11         maxi = max(maxi, len[i]);
12     return maxi;
13 }

```

4.5 Minimum Edit Distance

```

1  // 利用 dfs 輸出替換字串的步驟
2  void backtracking(int i, int j){
3      if(i == 0 || j == 0){
4          while(i > 0){
5              cout << cnt++ << " Delete " << i << endl;

```

```

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

```

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];

```

```

15     dfs(s1, 0, N/2, 0); // 折半枚舉
16     dfs(s2, N/2, N, 0);
17     long long int ans = 0;
18     // 題目:枚舉集合 Sx 的數字 Sxi，找出 Sy
19     // 集合內小於等於 T-Sxi 中最大的數 Syj
20     for(auto &x : s1){
21         auto it = s2.upper_bound(T - x);
22         long long int y = *(--it);
23         if(x + y <= T) ans = max(ans, x + y);
24     }
25     cout << ans << endl;

```

6 Graph

6.1 SPFA

```

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

```

6.2 Dijkstra

```

1  /* Dijkstra 最短路徑 */
2  struct Edge{
3      int v, w;
4  };
5  struct Item{
6      int u, dis;
7      // 取路徑最短
8      bool operator < (const Item &other) const{
9          return dis > other.dis;
10     }
11 };
12 int dis[maxn];
13 vector<Edge> G[maxn];
14 void dijkstra(int s){
15     for(int i = 0; i <= m; i++){
16         dis[i] = inf;
17     }
18     dis[s] = 0;
19     priority_queue<Item> pq;
20     pq.push({s, 0});
21     while(!pq.empty()){

```

```

22 // 取路徑最短的點
23 Item now = pq.top();
24 pq.pop();
25 if(now.dis > dis[now.u]){
26     continue;
27 }
28 // 把與 now.u 相連的點都跑一遍
29 for(Edge e : G[now.u]){
30     if(dis[e.v] > now.dis + e.w){
31         dis[e.v] = now.dis + e.w;
32         pq.push({e.v, dis[e.v]});
33     }
34 }
35 }
36 }

```

6.3 Floyd Warshall

```

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

```

6.4 Disjoint set Kruskal

```

1 struct Edge{
2     int u, v, 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     if(a != b){
21         if(parent[a] < parent[b]){
22             parent[a] += parent[b];
23             parent[b] = a;
24         }
25         else{
26             parent[b] += parent[a];
27             parent[a] = b;
28         }
29     }
30 }
31 void kruskal(){
32     memset(parent, -1, sizeof(parent));
33     sort(edge, edge + m);
34     int i, j;

```

```

35     for(i = 0, j = 0; i < n - 1 && j < m; i++){
36         // 如果 u 和 v 的祖先相同, 則 j++
37         // (祖先相同代表會產生環 所以不要)
38         while(find(edge[j].u) == find(edge[j].v)) j++;
39         // 若部會產生環 則讓兩點之間產生橋
40         // (連接兩顆子生成樹)
41         unite(edge[j].u, edge[j].v);
42         j++;
43     }
44 }

```

6.5 Disjoint set Kruskal 2

```

1 struct Edge{
2     int u, v;
3     double w;
4     bool operator < (const Edge &rhs) const{
5         return w < rhs.w;
6     }
7 }edge[maxn * maxn];
8 vector<Edge> G[maxn]; // 紀錄有哪些邊在 MST 上
9 int parent[maxn];
10 // disjoint set
11 int find(int x){
12     return x == parent[x] ? x : parent[x] = find(parent[x]);
13 }
14 bool unite(int a, int b){
15     int x = find(a);
16     int y = find(b);
17     if(x == y) return false;
18     parent[x] = y;
19     return true;
20 }
21 double kruskal(){
22     m = 0; // m: 邊的數量
23     for(int i = 0; i < n; ++i)
24         for(int j = i + 1; j < n; ++j)
25             edge[m++] = (Edge){i, j, dist(i, j)};
26     sort(edge, edge + m);
27     for(int i = 0; i < n; ++i){
28         parent[i] = i;
29         G[i].clear();
30     }
31     double total = 0.0;
32     int edge_cnt = 0;
33     for(int i = 0; i < m; ++i){
34         int u = edge[i].u, v = edge[i].v;
35         double cnt = edge[i].w;
36         if(unite(u, v)){
37             G[u].push_back((Edge){u, v, cnt});
38             G[v].push_back((Edge){v, u, cnt});
39             total += cnt;
40             if(++edge_cnt == n-1) break;
41         }
42     }
43     return total;
44 }

```

6.6 Bipartite

```

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     }
14 }

```

```

13         if(color[s] == color[it]){
14             return false;
15         }
16     }
17     return true;
18 }
19 void isBipartite(){
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         isBipartite();
44     }
45 }

```

6.7 Hungarian algorithm

```

1  /* 匈牙利演算法 */
2  const int maxn = 500+5;
3  int t, N, bn, gn, match[maxn];
4  bool visited[maxn];
5  vector<vector<int>> G(maxn);
6  struct People{
7      int h;
8      string music, sport;
9      People(){
10         People(int h, string music, string sport){
11             this->h = h;
12             this->music = music;
13             this->sport = sport;
14         }
15     }lef[maxn], rig[maxn];
16     bool check(People boy, People girl){
17         if(abs(boy.h - girl.h) <= 40 && boy.music ==
18             girl.music && boy.sport != girl.sport) return
19             true;
20         return false;
21     }
22     bool dfs(int s){
23         for(int i = 0; i < G[s].size(); ++i){
24             int v = G[s][i];
25             if(visited[v]) continue;
26             visited[v] = true;
27             if(match[v] == -1 || dfs(match[v])){
28                 match[v] = s;
29                 return true;
30             }
31         }
32         return false;
33     }
34     int Hungarian(){
35         int cnt = 0;
36         memset(match, -1, sizeof(match));
37         for(int i = 0; i < bn; ++i){
38             memset(visited, false, sizeof(visited));
39             if(dfs(i)) cnt++;
40         }
41     }

```

```

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 }

```

6.8 LCA

```

1  /*最低共同祖先*/
2  // 此 node 下有幾顆 node
3  int dfs(int node, int dep){
4      depth[node] = dep + 1;
5      if(G[node].empty()){
6          siz[node] = 1;
7          return 1;
8      }
9      int total = 1;
10     for(auto i : G[node])
11         total += dfs(i.v, dep + 1);
12     siz[node] = total;
13     return siz[node];
14 }
15 // 找出每個節點的 2^i 倍祖先
16 // 2^20 = 1e6 > 200000
17 void find_parent(){
18     for(int i = 1; i < 20; i++){
19         for(int j = 0; j < N; j++){
20             parent[j][i] =
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                 dif >>= 1;
32             }
33         }
34         if(a == b) return a;
35         for(int i = 19; i >= 0; i--){
36             if(parent[a][i] != parent[b][i]){
37                 a = parent[a][i];
38                 b = parent[b][i];
39             }
40         }
41         return parent[a][0];
42     }
43 }

```

6.9 Trie

```

1  /* Trie 字典樹 */
2  struct Tire{

```

```

3   int path;
4   map<string, int> G[maxn];
5   void init(){
6       path = 1;
7       G[0].clear();
8   }
9   void insert(string str){
10      int u = 0;
11      string word = "";
12      for(int i = 0; i < str.size(); ++i){
13          if(str[i] == '\\'){
14              if(!G[u].count(word)){
15                  G[path].clear();
16                  G[u][word] = path++;
17              }
18              u = G[u][word];
19              word = "";
20          }
21          else word += str[i];
22      }
23  }
24  void put(int u, int space){
25      for(auto i = G[u].begin(); i != G[u].end(); ++i){
26          for(int j = 0; j < space; ++j){
27              cout << " ";
28          }
29          cout << i->first << endl;
30          put(i->second, space + 1);
31      }
32  }
33 }tree;

```

7 Math

7.1 Hash

```

1  /* 建議搭配 Other - Stammering Aliens 食用 */
2  #define ull unsigned long long int
3  const int maxn = 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             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     }
52     long long int dist[maxn];
53     // 從 0 開始到每個點的距離
54     void distance(){
55         for (int u = 0; u < N; ++u){
56             for(int i = 0; i < G[u].size(); ++i){
57                 dist[G[u][i].v] = dist[u] + G[u][i].w;
58             }
59         }
60     }
61     int main(){
62         while(cin >> N && N){
63             memset(dist, 0, sizeof(dist));
64             memset(parent, 0, sizeof(parent));
65             memset(depth, 0, sizeof(depth));
66             memset(siz, 0, sizeof(siz));
67             for(int i = 0; i <= N; ++i){
68                 G[i].clear();
69             }
70             for(int i = 1; i < N; ++i){
71                 int u, w;
72                 cin >> u >> w;
73                 G[u].push_back({i, w});
74                 parent[i][0] = u;
75             }
76             find_parent();
77             dfs(0, 0);
78             distance();
79             int s; cin >> s;
80             bool space = false;
81             for(int i = 0; i < s; ++i){
82                 int a, b;
83                 cin >> a >> b;
84                 int lca = LCA(a, b);
85                 if(space) cout << " ";
86                 space = true;
87                 cout << (dist[a] + dist[b]) - (dist[lca]
88                     * 2);
89             }
90             cout << endl;
91         }
92     }
93 }

```

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

```

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         memset(exi, false, sizeof(exi));
35         memset(visited, false, sizeof(visited));
36         for(int i = 1; i <= n; ++i){
37             int siz; cin >> siz;
38             for(int j = 0; j < siz; ++j){
39                 int num; cin >> num;
40                 G[i].emplace_back(num);
41             }
42             cnt = 0;
43             dfs(1, 1);
44             if(n == 1) cnt++;
45             cout << cnt << endl;
46         }
47     }

```

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     }

```

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;
64 }

```

```

31 bool solve_y(){
32     memset(used, false, sizeof(used));
33     for(int i = 0; i < n; ++i){
34         y[i].pos = 0;
35         for(int j = y[i].lef; j <= y[i].rig; ++j){
36             if(!used[j]){
37                 y[i].pos = j;
38                 used[j] = true;
39                 break;
40             }
41         }
42         if(y[i].pos == 0) return false;
43     }
44     return true;
45 }
46 int main(){
47     while(cin >> n && n){
48         int x1, y1, x2, y2;
49         for(int i = 0; i < n; ++i){
50             cin >> x1 >> y1 >> x2 >> y2;
51             x[i].lef = min(x1, x2);
52             x[i].rig = max(x1, x2);
53             y[i].lef = min(y1, y2);
54             y[i].rig = max(y2, y2);
55             x[i].idx = y[i].idx = i;
56             x[i].pos = y[i].pos = 0;
57         }
58         sort(x, x + n);
59         sort(y, y + n);
60         if(!solve_x() || !solve_y()) cout <<
61             "IMPOSSIBLE" << endl;
62         else{
63             int ans_x[maxn], ans_y[maxn];
64             for(int i = 0; i < n; ++i){
65                 ans_x[x[i].idx] = x[i].pos;
66                 ans_y[y[i].idx] = y[i].pos;
67             }
68             for(int i = 0; i < n; ++i)
69                 cout << ans_x[i] << " " << ans_y[i]
70                     << endl;
71         }
72     }
73 }

```

8.6 Fabled Rooks

```

1  /* 特定排序後放入格子
2  以右邊大小排序 要從左邊開始放
3  以左邊大小排序 要從右邊開始放 */
4  int n;
5  const int maxn = 5000+5;
6  struct Edge{
7      int lef, rig, pos, idx;
8      bool operator < (const Edge &rhs) const{
9          if(rig != rhs.rig)
10             return rig < rhs.rig;
11         else
12             return lef < rhs.lef;
13     }
14 }x[maxn], y[maxn];
15 bool used[maxn];
16 bool solve_x(){
17     memset(used, false, sizeof(used));
18     for(int i = 0; i < n; ++i){
19         x[i].pos = 0;
20         for(int j = x[i].lef; j <= x[i].rig; ++j){
21             if(!used[j]){
22                 x[i].pos = j;
23                 used[j] = true;
24                 break;
25             }
26         }
27         if(x[i].pos == 0) return false;
28     }
29     return true;
30 }

```

8.7 Rails

```

1  /* deque 火車
2  倒退逆推法 能怎樣進來就能怎樣出去
3  lis: 1 2 3 4 5
4  dq: 3 2 4 1 5
5  1. 如果 lis front = dq front, dq pop
6  2. 反之 ans.push dq front, 每次檢查 ans top 是否 =
7     lis front
8  */
9  int main(){
10     int n;
11     while(cin >> n && n){
12         int train;
13         deque<int> dq;
14         while(cin >> train && train){
15             dq.emplace_back(train);
16             deque<int> lis, ans;
17             for(int i = 2; i <= n; ++i){
18                 cin >> train;
19                 dq.emplace_back(train);
20             }
21             for(int i = 1; i <= n; ++i)
22                 lis.emplace_back(i);
23             for(int i = 0, j = 0; j < n, i < n; ++i){
24                 if(lis[i] == dq[j]) ++j;
25                 else
26                     ans.emplace_back(lis[i]);
27                 while(!ans.empty()){
28                     if(dq[j] != ans.back()) break;
29                     ans.pop_back();
30                 }
31             }
32         }
33     }
34 }

```



```

29         ++j;
30     }
31 }
32 if(!ans.empty())
33     cout << "No" << endl;
34 else
35     cout << "Yes" << endl;
36 dq.clear();
37 }
38 cout << endl;
39 }
40 }

```

8.8 String Distance and Transform Process

```

1  /* MED - Minimum Edit Distance
2  增加刪除修改 使得字串A 以最小步驟數替換成 字串B
3  abcac
4  bcd
5      j 0 b c d
6  i +-----
7  0 | 0 1 2 3
8  a | 1 1 2 3
9  b | 2 1 2 3
10 c | 3 2 1 2
11 a | 4 3 2 2
12 c | 5 4 3 3
13 1 Delete 1
14 2 Replace 3,d
15 3 Delete 4 */
16 const int maxn = 80+5;
17 string strA, strB;
18 int dis[maxn][maxn];
19 int cnt;
20 // 利用 dfs 輸出替換字串的步驟
21 void backtracking(int i, int j){
22     if(i == 0 || j == 0){
23         while(i > 0){
24             cout << cnt++ << " Delete " << i << endl;
25             i--;
26         }
27         while(j > 0){
28             cout << cnt++ << " Insert " << i + 1 <<
29             ", " << strB[j-1] << endl;
30             j--;
31         }
32         return;
33     }
34     if(strA[i-1] == strB[j-1]){
35         backtracking(i-1, j-1);
36     }
37     else{
38         if(dis[i][j] == dis[i-1][j-1] + 1){
39             cout << cnt++ << " Replace " << i << ", "
40             << strB[j-1] << endl;
41             backtracking(i-1, j-1);
42         }
43         else if(dis[i][j] == dis[i-1][j] + 1){
44             cout << cnt++ << " Delete " << i << endl;
45             backtracking(i-1, j);
46         }
47         else if(dis[i][j] == dis[i][j-1] + 1){
48             cout << cnt++ << " Insert " << i + 1 <<
49             ", " << strB[j-1] << endl;
50             backtracking(i, j-1);
51         }
52     }
53 }
54 void MED(){
55     // 由於 B 是 0，所以 A 轉換成 B
56     // 時每個字元都要被刪除
57     for(int i = 0; i <= strA.size(); ++i) dis[i][0] =
58     i;
59     // 由於 A 是 0，所以 A 轉換成 B
60     // 時每個字元都需要插入

```

```

55     for(int j = 0; j <= strB.size(); ++j) dis[0][j] =
56     j;
57     for(int i = 1; i <= strA.size(); ++i){
58         for(int j = 1; j <= strB.size(); ++j){
59             // 字元相同代表不需修改，修改距離直接延續
60             if(strA[i-1] == strB[j-1]) dis[i][j] =
61             dis[i-1][j-1];
62             else{
63                 // 取 replace, delete, insert
64                 // 最小，選其 +1 為最少編輯距離
65                 dis[i][j] = min(dis[i-1][j-1],
66                 min(dis[i-1][j], dis[i][j-1])) +
67                 1;
68             }
69         }
70     }
71 }
72 int main(){
73     bool space = false;
74     while(getline(cin, strA) && getline(cin, strB)){
75         cnt = 1;
76         MED();
77         if(space) cout << endl;
78         space = true;
79         cout << dis[strA.size()][strB.size()] << endl;
80         backtracking(strA.size(), strB.size());
81     }
82 }

```

9 Greedy

9.1 Sticks

```

1  /* Greedy + dfs */
2  const int maxn = 100+5;
3  int n, stickLengthSum, ans, stick[maxn];
4  bool visited[maxn];
5  bool dfs(int length, int idx, int stickTotal){
6      if(length == ans){
7          if(stickTotal == n) return true;
8          length = 0;
9      }
10     if(length == 0){
11         for(idx = 0; visited[idx]; idx++){
12             visited[idx] = true;
13             if(dfs(length + stick[idx], idx+1,
14             stickTotal+1)) return true;
15             visited[idx] = false;
16         }
17     }
18     else{
19         for(int j = idx; j < n; ++j){
20             if(visited[j] || (j && stick[j] ==
21             stick[j-1] && !visited[j-1]))
22                 continue;
23             if(stick[j] + length > ans) continue;
24             visited[j] = true;
25             if(dfs(length + stick[j], j+1,
26             stickTotal+1)) return true;
27             visited[j] = false;
28             if(length + stick[j] == ans) return false;
29         }
30     }
31     return false;
32 }
33 int main(){
34     while(scanf("%d", &n) && n){
35         stickLengthSum = 0;
36         for(int i = 0; i < n; ++i){
37             scanf("%d", &stick[i]);
38             stickLengthSum += stick[i];
39         }
40         sort(stick, stick + n, greater<int>());
41         for(ans = stick[0]; ans <= stickLengthSum;
42         ans++){

```

```

37     memset(visited, false, sizeof(visited));
38     if(stickLengthSum % ans != 0) continue;
39     if(dfs(0, 0, 0)) break;
40 }
41 printf("%d\n", ans);
42 }
43 }

```

10 DP

10.1 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 3 6
7  0 2 2 2 2 3 3 3 3 5
8  0 1 1 2 2 3 3 3 3 4
9  因為這題可以重複使用同一條魔法
10 所以可以這樣dp */
11 int a[10000+5], b[10000+5];
12 int dp[10000+5][10000+5];
13 int main(){
14     int h, n;
15     cin >> h >> n;
16     for(int i = 1; i <= n; i++){
17         cin >> a[i] >> b[i];
18     }
19     memset(dp, 0x3f3f3f3f, sizeof(dp));
20     dp[0][0] = 0;
21     for(int i = 1; i <= n; i++){
22         for(int j = 0; j <= h; j++){
23             dp[i][j] = min(dp[i-1][j], dp[i][max(0, j
24             - a[i])] + b[i]);
25     }
26     cout << dp[n][h] << endl;
27 }

```

10.2 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 }

```

10.3 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 }

```

10.4 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 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 }

```

10.5 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;

```

```

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

```

10.6 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 }

```

10.7 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
   106 106 106 106 106 151 151
4  第二個人的負重: 20

```

```

5  0 0 0 0 52 52 52 52 52 54 54 54 54 106 106 106 106
   106 106 106 106
6  第三個人的負重: 20
7  0 0 0 0 52 52 52 52 52 54 54 54 54 106 106 106 106
   106 106 106 106
8  第四個人的負重: 26
9  0 0 0 0 52 52 52 52 52 54 54 54 54 106 106 106 106
   106 106 106 106 106 151 151 151 151 151 */
10 struct Edge{
11     int p;
12     int w;
13 }edge[1000+5];
14 int main(){
15     int t;
16     cin >> t;
17     while(t--){
18         int n; cin >> n;
19         for(int i = 0; i < n; i++)
20             cin >> edge[i].p >> edge[i].w;
21         int g, total = 0;
22         cin >> g;
23         for(int i = 0; i < g; i++){
24             int pw; cin >> pw;
25             int dp[30+5];
26             memset(dp, 0, sizeof(dp));
27             for(int j = 0; j < n; j++)
28                 for(int k = pw; k >= edge[j].w; k--)
29                     dp[k] = max(dp[k], dp[k -
30                         edge[j].w] + edge[j].p);
31             total += dp[pw];
32         }
33         cout << total << endl;
34     }
35 }

```

10.8 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 }

```

10.9 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 }

```

10.10 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 }

```

10.11 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 }

```

10.12 Stamps

```

1  /* dp - dfs/分配可能性並更新 */
2  const int maxn = 100+5;
3  int h, k, r, maxi = 0;
4  int x[maxn], y[maxn];
5  int ans[maxn]; // 存可貼出最大郵票值的面額
6  void dfs(int i){
7      // 若 x[i] 的 i 多於可貼的郵票數量
8      if(i >= k){
9          if(r > maxi){
10             maxi = max(maxi, r);
11             for(int i = 0; i < k; ++i)
12                 ans[i] = x[i];

```

```

13     }
14     return;
15 }
16 // 存此層尚未更新前的 r、y 值，因為 dfs
    完要回去上一層
17 int r_before_this_layer = r;
18 int y_before_this_layer[maxn];
19 for(int j = 0; j < maxn; ++j)
20     y_before_this_layer[j] = y[j];
21 // next: 下一可考慮的郵票面額
22 // postage: 貼完郵票的總面額(y的idx)
23 // num: 要貼幾張
24 // x[i-1] 要 -1 是因為 x 從 0 開始存第一種面額
25 for(int next = x[i-1] + 1; next <= r + 1; ++next){
26     x[i] = next;
27     for(int postage = 0; postage < x[i-1] * h;
        ++postage){
28         if(y[postage] >= h) continue;
29         for(int num = 1; num <= h - y[postage];
            ++num)
30             if(y[postage] + num < y[postage + num]
                * next) && (postage + num * next
                    < maxn))
31                 y[postage + num * next] =
                    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 }

```

10.13 Evacuation Plan

```

1 /* dp - 路徑/隊伍分配救難所 */
2 const int maxn = 4000+5;
3 int path[maxn][maxn];
4 long long int dp[maxn][maxn];
5 struct Edge{
6     int idx, position;
7     bool operator < (const Edge &rhs) const{
8         return position < rhs.position;
9     }
10 }team[maxn], shelter[maxn];
11 int main(){
12     int n;
13     while(cin >> n){
14         for(int i = 1; i <= n; ++i){
15             cin >> team[i].position;
16             team[i].idx = i;
17         }
18         sort(team + 1, team + n + 1);
19         int m; cin >> m;

```

```

20         for(int i = 1; i <= m; ++i){
21             cin >> shelter[i].position;
22             shelter[i].idx = i;
23         }
24         sort(shelter + 1, shelter + m + 1);
25         memset(dp, 0x3f3f3f3f, sizeof(dp));
26         dp[1][0] = 0;
27         for(int i = 1; i <= m; ++i){
28             for(int j = i; j <= n; ++j){
29                 // dp[i][j] = min(dp[i][j-1],
                    dp[i-1][j-1]) +
                    abs(team[j].position -
                        shelter[i].position);
30                 if(dp[i][j-1] <= dp[i-1][j-1]){
31                     dp[i][j] = min(dp[i][j-1],
                        dp[i-1][j-1]) +
                        abs(team[j].position -
                            shelter[i].position);
32                     path[i][j] = 0; //
                        從左邊來，前面的 teams 有人來
                        j shelter
33                 }
34                 else{
35                     dp[i][j] = min(dp[i][j-1],
                        dp[i-1][j-1]) +
                        abs(team[j].position -
                            shelter[i].position);
36                     path[i][j] = 1; //
                        從左上來，前面的 teams 不會來
                        j shelter
37                 }
38             }
39         }
40         int now_shelter = m;
41         int ans[maxn];
42         //
43         紀錄路徑，若從左邊來，上一隊也來此；若從右邊來，上隊
44         for(int i = n; i > 0; --i){
45             ans[team[i].idx] =
                shelter[now_shelter].idx;
46             now_shelter -= path[now_shelter][i];
47         }
48         cout << dp[m][n] << endl;
49         for(int i = 1; i < n; ++i)
50             cout << ans[i] << " ";
51         cout << ans[n] << endl;
52 }

```

10.14 Ladies Choice

```

1 /* dp - ladies & men */
2 const int maxn = 1000+5;
3 int n;
4 int man[maxn][maxn], manidx[maxn], lady[maxn][maxn],
    ladyidx[maxn];
5 int dp[maxn];
6 deque<int> dq;
7 void dp_func(){
8     while(!dq.empty()){
9         int man_now = dq.front();
10        dq.pop_front();
11        // manidx 現在指著的 lady
12        int lady1 = manidx[man_now];
13        // man 目前最想要的 lady
14        int lady_first = man[man_now][lady1];
15        // ladyidx 現在指著的 man
16        int man1 = ladyidx[lady_first];
17        if(man1 == 0){
18            dp[man_now] = lady_first;
19            ladyidx[lady_first] = man_now;
20        }
21        else if(lady[lady_first][man1] >
            lady[lady_first][man_now]){
22            dp[man_now] = lady_first;

```

```

23         manidx[man1]++;
24         dq.emplace_back(man1);
25         ladyidx[lady_first] = man_now;
26     }
27     else{
28         dq.emplace_back(man_now);
29         manidx[man_now]++;
30     }
31 }
32 int main(){
33     int t; cin >> t;
34     bool space = false;
35     while(t--){
36         cin >> n;
37         if(space) cout << endl;
38         space = true;
39         memset(man, 0, sizeof(man));
40         memset(lady, 0, sizeof(lady));
41         memset(manidx, 0, sizeof(manidx));
42         memset(ladyidx, 0, sizeof(ladyidx));
43         dq.clear();
44         for(int i = 1; i <= n; ++i){
45             for(int j = 1; j <= n; ++j)
46                 cin >> man[i][j];
47             dq.emplace_back(i);
48             manidx[i] = 1;
49         }
50         for(int i = 1; i <= n; ++i){
51             for(int j = 1; j <= n; ++j){
52                 int man_lady;
53                 cin >> man_lady;
54                 lady[i][man_lady] = j;
55             }
56         }
57         dp_func();
58         for(int i = 1; i <= n; ++i)
59             cout << dp[i] << endl;
60     }
61 }
62 }

```

11 LIS

11.1 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 }

```

11.2 Robots II

```

1  /* LIS
2  No.: 2 4 11 13 25 28 41 42
3  LIS: 1 2 3 4 4 5 5 5
4  num: 1 1 1 1 1 2 2 4
5  path: -1 0 1 2 2 3 3 5 */
6  const int maxn = 100+5;
7  int r, c;
8  vector<int> G;
9  int LIS[maxn * maxn], num[maxn * maxn], path[maxn *
10     maxn];
11 bool garbage[maxn][maxn];
12 void show_path(int n){
13     if(path[n] != -1) show_path(path[n]);
14     if((n != G.size() - 1) || garbage[r][c]) cout <<
15         " " << G[n];
16 }
17 int main(){
18     int ca = 1;
19     while(cin >> r >> c && (r != -1) && (c != -1)){
20         memset(garbage, false, sizeof(garbage));
21         G.clear();
22         int x, y;
23         while(cin >> x >> y && x && y){
24             garbage[x][y] = true;
25         }
26         // 紀錄有垃圾的點的編號
27         for(int i = 1; i <= r; ++i){
28             for(int j = 1; j <= c; ++j){
29                 if(garbage[i][j]) G.emplace_back((i -
30                     1) * c + j);
31             }
32         }
33         // 如果終點沒有垃圾，假設他有
34         if(!garbage[r][c]) G.emplace_back(r * c);
35         G.emplace_back(0);
36         // i 和 j
37         // 是按照編號大小順序由小排到大的垃圾編號
38         for(int i = 0; i < G.size(); ++i){
39             LIS[i] = 1;
40             num[i] = 1;
41             path[i] = -1;
42             for(int j = 0; j < i; ++j){
43                 // 判斷垃圾的 col 前後
44                 if(((G[j] - 1) % c) <= ((G[i] - 1) %
45                     c)){
46                     // num 是經過的路徑數量。path
47                     // 是從誰來
48                     if(LIS[i] == LIS[j] + 1){

```

```

43         num[i] += num[j];
44     }
45     else if(LIS[i] < LIS[j] + 1){
46         LIS[i] = LIS[j] + 1;
47         num[i] = num[j];
48         path[i] = j;
49     }
50 }
51 }
52 }
53 G.pop_back();
54 // 要把假設還回去
55 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 }

```

12 Math

12.1 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

```

12.2 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     }
22 }

```

12.3 Fraction Floor Sum

```

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

```

12.4 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 }

```

12.5 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 }

```

12.6 ORXOR

```

1  /* bitwise operator 二進位制數論
2  如何切區段，之所以要1<<n是為了可以跑000~111
3  i = 0, binary i = 000
4  0 : 1 5 7
5  i = 1, binary i = 001
6  1 : 1 5 7
7  i = 2, binary i = 010, 看得出來切了一刀
8  2 : 1 | 5 7
9  i = 3, binary i = 011
10 3 : 1 | 5 7
11 i = 4, binary i = 100, 為了要切在index=2, 所以才要1<<j
12 4 : 1 5 | 7
13 i = 5, binary i = 101
14 5 : 1 5 | 7
15 i = 6, binary i = 110
16 6 : 1 | 5 | 7
17 i = 7, binary i = 111
18 7 : 1 | 5 | 7
19 可以觀察出來，前兩位 bit 是 1 時代表的意義是切在哪裡
20 */
21 int main(){
22     int n; cin >> n;
23     int num[20+7];
24     memset(num, 0, sizeof(num));
25     for(int i = 1; i <= n; i++){
26         cin >> num[i];
27         // 不知道為什麼只有 2147483647 給過
28         int mini = 2147483647;
29         // 1 << n = n * 2
30         for(int i = 0; i < (1 << n); i++){
31             int XOR = 0, OR = 0;
32             for(int j = 1; j <= n; j++){
33                 OR |= num[j];
34                 if((i & (1 << j))){
35                     XOR ^= OR;
36                     OR = 0;
37                 }
38             }
39             XOR ^= OR;
40             mini = min(mini, XOR);
41         }
42     }
43     cout << mini << endl;
44 }

```

12.7 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 }

```

12.8 Playing With Stones

```

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

```

12.9 And Then There Was One

```

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

```

13 Binary Search

13.1 Fill the Containers

```

1  /*binary search 變形*/
2  int binary_search(int arr[maxn], int lef, int rig,
3      int mini){
4      if(lef > rig) return mini;
5      int amount = 1, fill = 0;
6      int mid = (lef + rig) >> 1;
7      for(int i = 0; i < n; ++i){
8          if(amount > m) break;
9          fill += arr[i];
10         if(fill > mid){
11             fill = arr[i];
12             amount++;
13         }
14     }
15     if(!flag && amount <= m) mini = mid;
16     if(flag && amount == m) mini = mid;
17     if(amount == m){

```



```

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     }

```

13.2 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 }

```

14 Graph

14.1 Maximum sum on a torus

```

1  /* Prefix sum in Graph*/
2  const int maxn = 80;
3  const int inf = 0x3f3f3f3f;
4  int arr[maxn*2 + 5][maxn*2 + 5];
5  int prefix_sum[maxn*2 + 5][maxn*2 + 5];
6  int ans[maxn*2];
7  int n;
8  int maxSub(int start){
9      int maxi, dp;
10     maxi = dp = ans[start];
11     for(int i = start + 1; i < start + n; ++i){
12         dp += ans[i];
13         maxi = max(maxi, dp);
14     }

```

```

15     return maxi;
16 }
17 int main(){
18     int t;
19     cin >> t;
20     while(t--){
21         memset(arr, 0, sizeof(arr));
22         cin >> n;
23         for(int i = 0; i < n; ++i)
24             for(int j = 0; j < n; ++j){
25                 cin >> arr[i][j];
26                 arr[n+i][j] = arr[i][n+j] =
27                     arr[n+i][n+j] = arr[i][j];
28             }
29         int len = 2*n;
30         memset(prefix_sum, 0, sizeof(prefix_sum));
31         for(int i = 0; i < len; ++i)
32             for(int j = 0; j < len; ++j){
33                 if(i == 0) prefix_sum[i][j] =
34                     arr[i][j];
35                 else prefix_sum[i][j] =
36                     prefix_sum[i-1][j] + arr[i][j];
37             }
38         int maxi = -inf;
39         for(int i = 0; i < len; ++i){
40             for(int j = i; j < i + n && j < len; ++j){
41                 for(int k = 0; k < len; ++k){
42                     if(i == 0) ans[k] =
43                         prefix_sum[j][k];
44                     else ans[k] = prefix_sum[j][k] -
45                         prefix_sum[i-1][k];
46                 }
47                 for(int k = 0; k < n; ++k){
48                     int answer = maxSub(k);
49                     maxi = max(maxi, answer);
50                 }
51             }
52         }
53         cout << maxi << endl;
54     }
55 }

```

15 Segement Tree

15.1 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[lel].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 segment 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         // 更新左邊界至被切區塊的右邊界加一，就不會切到區
44         lef = num[lef].rig + 1;
45     }
46     // 查詢的右區間邊界切到區塊，且此區間有數個區塊
47     if(rig < num[rig].rig){
48         // 計算切到的區間大小，並找出最大
49         ans = max(ans, rig - num[rig].lef + 1);
50         // 更新右邊界
51         rig = num[rig].lef - 1;
52     }
53     // 如果左邊界大於右邊界，表示不需要再進行查詢直接回傳
54     if(lef > rig) return ans;
55     if(tree[x].lef >= lef && tree[x].rig <= rig)
56         return tree[x].value;
57     int mid = (tree[x].lef + tree[x].rig) >> 1;
58     if(lef <= mid) ans = max(ans, query(lef, rig,
59         Lson(x)));
60     if(mid < rig) ans = max(ans, query(lef, rig,
61         Rson(x)));
62     return ans;
63 }
64 int main(){
65     int n, q;
66     while(cin >> n && n){
67         cin >> q;
68         int start = 1;
69         for(int i = 1; i <= n; ++i){
70             cin >> num[i].value;
71             if(num[i].value != num[i-1].value){
72                 for(int j = start; j < i; ++j){
73                     num[j].rig = i - 1;
74                     num[j].fre = i - start;
75                 }
76                 start = num[i].lef = i;
77             }
78             else num[i].lef = start;
79         }
80         // 最後一段 [start, n]
81         for(int j = start; j <= n; ++j){
82             num[j].rig = n;
83             num[j].fre = n - start + 1;
84         }
85         build(1, n, 1);
86         int lef, rig;
87         for(int i = 0; i < q; ++i){
88             cin >> lef >> rig;
89             cout << query(lef, rig, 1) << endl;
90         }
91     }
92 }

```

16 Dijkstra

16.1 Airport Express

```

1 /* Dijkstra 捷徑票 */
2 int n, m, S, T;

```

```

3 const int inf = 1e9;
4 const int maxn = 20000 + 5;
5 struct Edge{
6     int v, w;
7 };
8 struct Item{
9     int u, dis;
10    // 取路徑最短
11    bool operator < (const Item &other) const{
12        return dis > other.dis;
13    }
14 };
15 int dis[maxn], from[maxn];
16 vector<Edge> G[maxn];
17 void dijkstra(int s){
18     for(int i = 0; i <= n; i++){
19         dis[i] = inf;
20     }
21     dis[s] = 0;
22     for(int i = 0; i <= n; i++){
23         from[i] = i;
24     }
25     priority_queue<Item> pq;
26     pq.push({s, 0});
27     while(!pq.empty()){
28         // 取路徑最短的點
29         Item now = pq.top();
30         pq.pop();
31         if(now.dis > dis[now.u])
32             continue;
33         // 鬆弛 更新
34         // 把與 now.u 相連的點都跑一遍
35         for(Edge e : G[now.u]){
36             if(dis[e.v] > now.dis + e.w){
37                 dis[e.v] = now.dis + e.w;
38                 from[e.v] = now.u;
39                 pq.push({e.v, dis[e.v]});
40             }
41         }
42     }
43 }
44 deque<int> ans;
45 void dfs(int T){
46     ans.emplace_back(T);
47     if(from[T] != T) dfs(from[T]);
48 }
49 int main(){
50     bool space = false;
51     while(cin >> n >> S >> T){
52         if(!space) space = true;
53         else cout << endl;
54         for(int i = 0; i <= n; i++){
55             G[i].clear();
56         }
57         ans.clear();
58         cin >> m;
59         int u, v, w;
60         for(int i = 0; i < m; i++){
61             cin >> u >> v >> w;
62             // 無向圖
63             G[u].push_back({v, w});
64             G[v].push_back({u, w});
65         }
66         dijkstra(S);
67         dfs(T);
68         int ori = dis[T];
69         int mini = dis[T], state = 0;
70         int ticket;
71         cin >> ticket;
72         for(int i = 0; i < ticket; ++i){
73             cin >> u >> v >> w;
74             G[u].push_back({v, w});
75             dijkstra(S);
76             if(dis[T] < mini){
77                 mini = min(mini, dis[T]);
78                 state = u;
79                 ans.clear();
80                 dfs(T);
81             }
82             G[u].pop_back();
83         }
84     }
85 }

```

```

80     G[v].push_back({u, w});
81     dijkstra(S);
82     if(dis[T] < mini){
83         mini = min(mini, dis[T]);
84         state = v;
85         ans.clear();
86         dfs(T);
87     }
88     G[v].pop_back();
89 }
90 for(int i = ans.size()-1; i > 0; i--){
91     cout << ans[i] << " ";
92     cout << ans[0];
93     cout << endl;
94     if(mini == ori)
95         cout << "Ticket Not Used" << endl;
96     else
97         cout << state << endl;
98     cout << mini << endl;
99 }
100 }

```

16.2 Walk Through the Forest

```

1  /* Dijkstra + 路徑最優化 DP */
2  const int inf = 0x3f3f3f3f;
3  const int maxn = 1000+5;
4  int n, m;
5  struct Edge{
6      int v, w;
7  };
8  struct Item{
9      int u, dis;
10     bool operator < (const Item &other) const{
11         return dis > other.dis;
12     }
13 };
14 int dis[maxn];
15 long long int dp[maxn];
16 vector<Edge> G[maxn];
17 vector<int> path[maxn];
18 void dijkstra(int s){
19     for(int i = 0; i <= n; ++i){
20         dis[i] = inf;
21     }
22     dis[s] = 0;
23     priority_queue<Item> pq;
24     pq.push({s, 0});
25     while(!pq.empty()){
26         Item now = pq.top();
27         pq.pop();
28
29         if(now.dis > dis[now.u]){
30             continue;
31         }
32
33         for(Edge e: G[now.u]){
34             if(dis[e.v] > now.dis + e.w){
35                 dis[e.v] = now.dis + e.w;
36                 pq.push({e.v, dis[e.v]});
37             }
38         }
39     }
40 }
41 long long int dfs(int u){
42     // ans 是 pointer, 指向 dp[u] 的記憶體位址
43     // 對於 ans 的 value 改變會記錄在 dp[u]
44     long long int &ans = dp[u];
45     if(ans != -1) return ans;
46     if(u == 2) return ans = 1;
47     ans = 0;
48     for(int i = 0; i < path[u].size(); ++i)
49         ans += dfs(path[u][i]);
50     return ans;
51 }

```

```

52 int main(){
53     while(cin >> n && n){
54         cin >> m;
55         for(int i = 0; i <= n; ++i) G[i].clear();
56         int u, v, w;
57         for(int i = 0; i < m; ++i){
58             cin >> u >> v >> w;
59             G[u].push_back({v, w});
60             G[v].push_back({u, w});
61         }
62         dijkstra(2); // dijkstra
63         // 紀錄從終點到每個點的距離
64         memset(dp, -1, sizeof(dp));
65         for(int i = 1; i <= n; ++i){
66             path[i].clear();
67             for(int j = 0; j < G[i].size(); ++j){
68                 int v = G[i][j].v;
69                 // 如果到 v 的距離比到 i
70                 // 遠, 代表從起點經過 i 再到 v
71                 if(dis[i] > dis[v])
72                     path[i].push_back(v);
73             }
74             cout << dfs(1) << endl;
75         }
76     }
77 }

```

17 Kruskal

17.1 Qin Shi Huang Road System

```

1  /* kruskal disjoint set dfs */
2  const int maxn = 1000 + 5;
3  int n, m;
4  int x[maxn], y[maxn], p[maxn];
5  struct Edge{
6      int u, v;
7      double w;
8      bool operator < (const Edge &rhs) const{
9          return w < rhs.w;
10     }
11 }edge[maxn * maxn];
12 vector<Edge> G[maxn];
13 int parent[maxn];
14 // 計算兩點之間的距離
15 double dist(int a, int b){
16     double x2 = (x[a] - x[b]) * (x[a] - x[b]);
17     double y2 = (y[a] - y[b]) * (y[a] - y[b]);
18     return sqrt(x2 + y2);
19 }
20 // disjoint set
21 int find(int x){
22     return x == parent[x] ? x : parent[x] =
23         find(parent[x]);
24 }
25 bool unite(int a, int b){
26     int x = find(a);
27     int y = find(b);
28     if(x == y) return false;
29     parent[x] = y;
30     return true;
31 }
32 double kruskal(){
33     m = 0; // m: 邊的數量
34     for(int i = 0; i < n; ++i)
35         for(int j = i + 1; j < n; ++j)
36             edge[m++] = (Edge){i, j, dist(i, j)};
37     sort(edge, edge + m);
38     for(int i = 0; i < n; ++i){
39         parent[i] = i;
40         G[i].clear();
41     }
42     double total = 0.0;
43     int edge_cnt = 0;

```

```

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

```

18 Bipartite Graph

18.1 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

```

```

16         return false;
17     }
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 }

```

18.2 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));

```

```

40     for(int i = 0; i < bn; ++i){
41         memset(visited, false, sizeof(visited));
42         if(dfs(i)) cnt++;
43     }
44     return cnt;
45 }
46 int main(){
47     int t;
48     cin >> t;
49     while(t--){
50         int N;
51         cin >> N;
52         bn = 0, gn = 0;
53         for(int i = 0; i <= N; ++i) G[i].clear();
54         int h;
55         string sex, music, sport;
56         for(int i = 0; i < N; ++i){
57             cin >> h >> sex >> music >> sport;
58             if(sex == "M")
59                 lef[bn++] = People(h, music, sport);
60             else
61                 rig[gn++] = People(h, music, sport);
62         }
63         for(int i = 0; i < bn; ++i)
64             for(int j = 0; j < gn; ++j)
65                 if(check(lef[i], rig[j]))
66                     G[i].emplace_back(j);
67         cout << N - Hungarian() << endl;
68     }
69 }

```

18.3 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     meset(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(){

```

```

41     int t;
42     scanf("%d", &t);
43     while(t--){
44         scanf("%d", &n);
45         for(int i = 0; i < n; ++i) G[i].clear();
46         for(int i = 0; i < n; ++i){
47             int h, m;
48             scanf("%d:%d", &h, &m);
49             p[i].s = h * 60 + m;
50             scanf("%d%d%d%d", &p[i].x1, &p[i].y1,
51                     &p[i].x2, &p[i].y2);
52         }
53         sort(p, p + n);
54         for(int i = 0; i < n; ++i)
55             for(int j = i + 1; j < n; ++j)
56                 if(check(p[i], p[j]))
57                     G[i].push_back(j);
58         printf("%d\n", n - Hungarian());
59     }

```

18.4 SAM I AM

```

1  /* 二分圖匹配 + 最小點覆蓋 */
2  const int maxn = 1000+5;
3  int R, C, N;
4  bool arr[maxn][maxn], visitX[maxn], visitY[maxn];
5  int matchX[maxn], matchY[maxn];
6  int dfs(int x){
7      visitX[x] = true;
8      for(int y = 1; y <= C; ++y){
9          if(arr[x][y] && !visitY[y]){
10             visitY[y] = true;
11             if(matchY[y] == 0 || dfs(matchY[y])){
12                 matchX[x] = y;
13                 matchY[y] = x;
14                 return 1;
15             }
16         }
17     }
18     return 0;
19 }
20 int Match(){
21     int sum = 0;
22     memset(matchX, 0, sizeof(matchX));
23     memset(matchY, 0, sizeof(matchY));
24     for(int i = 1; i <= R; ++i){
25         memset(visitX, false, sizeof(visitX));
26         memset(visitY, false, sizeof(visitY));
27         sum += dfs(i);
28     }
29     return sum;
30 }
31 int main(){
32     while(cin >> R >> C >> N && R && C && N){
33         memset(arr, false, sizeof(arr));
34         memset(visitX, false, sizeof(visitX));
35         memset(visitY, false, sizeof(visitY));
36         int row, col;
37         for(int i = 0; i < N; ++i){
38             cin >> row >> col;
39             arr[row][col] = true;
40         }
41         int cnt = Match();
42         cout << cnt;
43         memset(visitX, 0, sizeof(visitX));
44         memset(visitY, 0, sizeof(visitY));
45         for(int i = 1; i <= R; ++i){
46             if(matchX[i] == 0) dfs(i);
47             for(int i = 1; i <= R; ++i)
48                 if(!visitX[i]) cout << " r" << i;
49             for(int i = 1; i <= C; ++i)
50                 if(visitY[i]) cout << " c" << i;
51             cout << endl;
52         }
53     }

```

19 Function

19.1 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()

```

19.2 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 }

```

19.3 setprecision

```

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

```

19.4 GCD LCM

```

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

```

19.5 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

```

19.6 sort

```

1 priority_queue<int, vector<int>, less<int>> // 大到小
2 priority_queue<int, vector<int>, greater<int>> //
3     小到大
4 int arr[] = {4, 5, 8, 3, 7, 1, 2, 6, 10, 9};
5 sort(arr, arr+10);
6
7 vector<int> v;
8 sort(v.begin(), v.end()); //小到大
9
10 int cmp(int a, int b){
11     return a > b;
12 }
13 sort(v.begin(), v.end(), cmp); //大到小

```

19.7 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 }

```

19.8 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 {

```

```
10     cout << "Not found." << endl;
11 }
12 // cout: Found: 6
13
14 // 取值：使用 iterator
15 x = *st.begin(); // set 中的第一個元素(最小的元素)
16 x = *st.rbegin(); // set
    中的最後一個元素(最大的元素)
17
18 // search
19 iter = st.find(6);
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 }
```