

## Contents

|     |                        |   |
|-----|------------------------|---|
| 1   | Sync                   | 1 |
| 1.1 | Sync                   | 1 |
| 2   | Data Structure         | 1 |
| 2.1 | Binary Search          | 1 |
| 2.2 | BIT                    | 1 |
| 2.3 | BWT                    | 1 |
| 3   | Divide and Conquer     | 1 |
| 3.1 | count inversions       | 1 |
| 4   | DP                     | 1 |
| 4.1 | Doubling               | 1 |
| 4.2 | LCS                    | 2 |
| 4.3 | LIS                    | 2 |
| 4.4 | LIS 2                  | 2 |
| 4.5 | Minimum Edit Distance  | 2 |
| 5   | Enumerate              | 3 |
| 5.1 | Halfcut Enumerate      | 3 |
| 6   | Graph                  | 3 |
| 6.1 | SPFA                   | 3 |
| 6.2 | Dijkstra               | 3 |
| 6.3 | Floyd Warshall         | 3 |
| 6.4 | Disjoint set Kruskal   | 3 |
| 6.5 | Disjoint set Kruskal 2 | 4 |
| 6.6 | Bipartite              | 4 |
| 6.7 | Hungarian algorithm    | 4 |
| 6.8 | LCA                    | 5 |
| 6.9 | Trie                   | 5 |
| 7   | Math                   | 5 |
| 7.1 | Hash                   | 5 |
| 8   | Function               | 6 |
| 8.1 | CHAR                   | 6 |
| 8.2 | string                 | 6 |
| 8.3 | setprecision           | 6 |
| 8.4 | GCD LCM                | 6 |
| 8.5 | reverse                | 6 |
| 8.6 | sort                   | 6 |
| 8.7 | map                    | 6 |
| 8.8 | set                    | 6 |
| 9   | Other                  | 7 |
| 9.1 | Ants Colony            | 7 |
| 9.2 | Binary codes           | 7 |
| 9.3 | Disk Tree              | 7 |

## 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        }
19        k++;
20    }
21    for(int k = lef; k < rig; ++k) arr[k] = buf[k];
22    return ans;
23 }
```

## 4 DP

### 4.1 Doubling

```

1  /* 倍增 */
2  int LOG = sqrt(N); // 2^LOG >= N
3  vector<int> arr(N);
4  vector<vector<int>> dp(N, vector<int>(LOG));
5  for(int i = 0; i < N; ++i) cin >> arr[i];
6  int L, Q, a, b;
7  cin >> L >> Q;
8  for(int i = 0; i < N; ++i){
9      dp[i][0] = lower_bound(arr.begin(), arr.end(),
10         arr[i] + L) - arr.begin();
11      if(dp[i][0] == N || arr[i] + L < arr[dp[i][0]])
12         dp[i][0] -= 1;
13  }
14  for(int i = 1; i < LOG; ++i)
15      for(int j = 0; j < N; ++j)
16         dp[j][i] = dp[dp[j][i-1]][i-1];
17  for(int i = 0; i < Q; ++i){
18      cin >> a >> b;
19      a--; // 要減減是因為arr的index從0開始但題目從1開始
20      b--;
21      if(a > b) swap(a, b);
22      int ans = 0;
23      for(int i = LOG - 1; i >= 0; --i){ // 從後往回推
24          if(dp[a][i] < b){
25              ans += (1 << i);
26              a = dp[a][i];
27          }
28      }
29      cout << ans + 1 << endl;
30  }

```

## 4.2 LCS

```

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

```

## 4.3 LIS

```

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

```

## 4.4 LIS 2

```

1  int LIS(vector<int> &a){
2      int len[a.size()];
3      for(int i = 0; i < a.size(); ++i) len[i] = 1;

```

```

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

```

## 4.5 Minimum Edit Distance

```

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

```

## 5 Enumerate

### 5.1 Halfcut Enumerate

```

1  /* 折半枚舉 */
2  void dfs(set<long long int> &s, int depth, int T,
3         long long int sum){
4      if(depth >= T){
5          s.insert(sum);
6          return;
7      }
8      dfs(s, depth + 1, T, sum); // 取或不取的概念
9      dfs(s, depth + 1, T, sum + A[depth]);
10 }
11 int main(){
12     int N, T;
13     set<long long int> s1, s2;
14     cin >> N >> T;
15     for(int i = 0; i < N; ++i) cin >> A[i];
16     dfs(s1, 0, N/2, 0); // 折半枚舉
17     dfs(s2, N/2, N, 0);
18     long long int ans = 0;
19     // 題目:枚舉集合 Sx 的數字 Sxi, 找出 Sy
20     // 集合內小於等於 T-Sxi 中最大的數 Syj
21     for(auto &x : s1){
22         auto it = s2.upper_bound(T - x);
23         long long int y = *(--it);
24         if(x + y <= T) ans = max(ans, x + y);
25     }
26     cout << ans << endl;
27 }

```

## 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] =
13         find(parent[x]);
14 }
15 bool unite(int a, int b){
16     int x = find(a);
17     int y = find(b);
18     if(x == y) return false;
19     parent[x] = y;
20     return true;
21 }
22 double kruskal(){
23     m = 0; // m: 邊的數量
24     for(int i = 0; i < n; ++i)
25         for(int j = i + 1; j < n; ++j)
26             edge[m++] = (Edge){i, j, dist(i, j)};
27     sort(edge, edge + m);
28     for(int i = 0; i < n; ++i){
29         parent[i] = i;
30         G[i].clear();
31     }
32     double total = 0.0;
33     int edge_cnt = 0;
34     for(int i = 0; i < m; ++i){
35         int u = edge[i].u, v = edge[i].v;
36         double cnt = edge[i].w;
37         if(unite(u, v)){
38             G[u].push_back((Edge){u, v, cnt});
39             G[v].push_back((Edge){v, u, cnt});
40             total += cnt;
41             if(++edge_cnt == n-1) break;
42         }
43     }
44 }

```

```

41     }
42 }
43 return total;
44 }

```

## 6.6 Bipatirate

```

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

```

## 6.7 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         }
11     People(int h, string music, string sport){
12         this->h = h;
13         this->music = music;
14         this->sport = sport;
15     }
16 }lef[maxn], rig[maxn];
17 bool check(People boy, People girl){
18     if(abs(boy.h - girl.h) <= 40 && boy.music ==
19         girl.music && boy.sport != girl.sport) return
20         true;
21 }

```

```

18     return false;
19 }
20 bool dfs(int s){
21     for(int i = 0; i < G[s].size(); ++i){
22         int v = G[s][i];
23         if(visited[v]) continue;
24         visited[v] = true;
25         if(match[v] == -1 || dfs(match[v])){
26             match[v] = s;
27             return true;
28         }
29     }
30     return false;
31 }
32 int Hungarian(){
33     int cnt = 0;
34     memset(match, -1, sizeof(match));
35     for(int i = 0; i < bn; ++i){
36         memset(visited, false, sizeof(visited));
37         if(dfs(i)) cnt++;
38     }
39     return cnt;
40 }
41 int main(){
42     cin >> t;
43     while(t--){
44         cin >> N;
45         bn = 0, gn = 0;
46         for(int i = 0; i <= N; ++i) G[i].clear();
47         int h;
48         string sex, music, sport;
49         for(int i = 0; i < N; ++i){
50             cin >> h >> sex >> music >> sport;
51             if(sex == "M") lef[bn++] = People(h,
52                 music, sport);
53             else rig[gn++] = People(h, music, sport);
54         }
55         for(int i = 0; i < bn; ++i){
56             for(int j = 0; j < gn; ++j)
57                 if(check(lef[i], rig[j]))
58                     G[i].emplace_back(j);
59         }
60         cout << N - Hungarian() << endl;
61     }
62 }

```

## 6.8 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];

```

```

27         for (int i = 0; i < 20; i++){
28             if (dif & 1) b = parent[b][i];
29             dif >>= 1;
30         }
31     }
32     if (a == b) return a;
33     for (int i = 19; i >= 0; i--){
34         if (parent[a][i] != parent[b][i]){
35             a = parent[a][i];
36             b = parent[b][i];
37         }
38     }
39     return parent[a][0];
40 }

```

## 6.9 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 Function

### 8.1 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()

```

### 8.2 string

```

1 int main(){
2     string str;
3     while(cin >> str){
4         // substr 取 str idx 2~4 的值
5         cout << str.substr(2, 4) << endl;
6         // substr 取 str idx 2 以後的所有值
7         cout << str.substr(2) << endl;
8
9         string subst;
10        cin >> subst;
11        // str.append 連接字串
12        cout << str.append(subst) << endl;
13
14        char s[100], ss[100];
15        cin >> s >> ss;
16
17        char *p;
18        // strstr 回傳在s裡找到ss後的整個字串(從 ss
19        // idx 0 到結束)
20        p = strstr(s, ss);
21        cout << p << endl;
22        // strstr 也可以單純用來找字串
23        if(p != NULL) cout << "yes" << endl;
24        else cout << "no" << endl;
25    }
26 }

```

### 8.3 setprecision

```

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

```

### 8.4 GCD LCM

```

1 int gcd(int a, int b){
2     return (b == 0 ? a : gcd(b, a % b));
3 }
4 int lcm(int a, int b){
5     return a * b / gcd(a, b);
6 }
7
8 /* 輾轉相除法 - 求兩數是否互質
9 如果兩數互質 最終結果其中一方為0時 另一方必為1
10 若兩數有公因數 最終結果其中一方為0時 另一方必不為1 */
11 while ( ( num1 % num2 ) != 0 && ( num2 % num1 ) !=
12         0 );

```

### 8.5 reverse

```

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

```

### 8.6 sort

```

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

```

### 8.7 map

```

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

```

### 8.8 set

```

1 int main(){
2     set<int> st {1, 6, 8}; // 直接初始化的寫法
3     st.insert(1); // 也可以這樣寫就好
4     set<int>::iterator iter;
5
6     // 如果有找到，就會傳回正確的 iterator，否則傳回
7     // st.end()
8     if (iter != st.end()) {
9         cout << "Found: " << *iter << endl;
10    } else {

```

```

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 }

```

## 9 Other

### 9.1 Ants Colony

```

1  /* LCA 最低共同祖先 */
2  const int maxn = 1e5 + 5;
3  struct Edge{
4      int v;
5      int w;
6  };
7  int N;
8  vector<Edge> G[maxn];
9  int parent[maxn][20+5];
10 int depth[maxn], siz[maxn];
11 // 此 node 下有幾顆 node
12 int dfs(int node, int dep){
13     depth[node] = dep + 1;
14     if(G[node].empty()){
15         siz[node] = 1;
16         return 1;
17     }
18     int total = 1;
19     for(auto i : G[node])
20         total += dfs(i.v, dep + 1);
21     siz[node] = total;
22     return siz[node];
23 }
24 // 找出每個節點的 2^i 倍祖先
25 // 2^20 = 1e6 > 200000
26 void find_parent(){
27     for(int i = 1; i < 20; i++){
28         for (int j = 0; j < N; j++){
29             parent[j][i] =
                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 }

```

```

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

```

### 9.2 Binary codes

```

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

```

### 9.3 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[path].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 }tree;
34 int main(){
35     int n;
36     string str;
37     while(cin >> n && n){
38         tree.init();
39         for(int i = 0; i < n; ++i){
40             cin >> str;
41             str += '\\';
42             tree.insert(str);
43         }
44         tree.put(0, 0);
45         cout << endl;
46     }
47 }
```