Contents

1 Sync

1.1 Sync

```cpp
int main(){
    std::ios::sync_with_stdio(false);
    // 開始寫程式
}
```

2 Data Structure

2.1 Binary Search

```cpp
int binary_search(int arr[maxn], int lef, int rig,
    int target){
    if(lef > rig) return 0x3f3f3f3f;
    int mid = (lef + rig) >> 1;
    if(arr[mid] == target) return mid;
    else if(arr[mid] > target){
        return binary_search(arr, lef, mid - 1,
            target);
    }
    else{
        return binary_search(arr, mid + 1, rig,
            target);
    }
}
```

2.2 BIT

```cpp
/* BIT Binary Index Tree */
#define lowbit(k) (k & -k)
void add(vector<int> &tr, int id, int val) {
  for (; id <= n; id += lowbit(id)) {
    tr[id] += val;
  }
}
int sum(vector<int> &tr, int id) {
  int ret = 0;
  for (; id >= 1; id -= lowbit(id)) {
    ret += tr[id];
  }
  return ret;
}
```

2.3 BWT

```cpp
/* BWT 資料轉換演算法 */
void BWT(){
    for(int i = 0; i < n; ++i){
        if(back[i] == 0)
            mini[zero++] = i;
    for(int i = 0; i < n; ++i)
        if(back[i] == 1)
            mini[zero++] = i;
    int ptr = mini[0];
    for(int i = 0; i < n; ++i){
        cout << back[ptr] << " ";
        ptr = mini[ptr];
    }
    cout << endl;
}
```

3 Divide and Conquer

3.1 count inversions

```cpp
/*逆序數對*/
int arr[maxn], buf[maxn];
int count_inversions(int lef, int rig){
    if(rig - lef <= 1) return 0;
    int mid = (lef + rig)/2;
    int ans = count_inversions(lef, mid) +
        count_inversions(mid, rig);
    int i = lef, j = mid, k = lef;
    while(i < mid || j < rig){
        if(i >= mid) buf[k] = arr[j++];
```

```
10        else if(j >= rig) buf[k] = arr[i++];
11        else{
12            if(arr[i] <= arr[j]) buf[k] = arr[i++];
13            else{
14                buf[k] = arr[j++];
15                ans += mid - i;
16            }
17        }
18        k++;
19    }
20    for(int k = lef; k < rig; ++k) arr[k] = buf[k];
21    return ans;
22 }
```

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

4.2 LCS

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

4.3 LIS

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

4.4 LIS 2

```
1 int LIS(vector<int> &a){
2     int len[a.size()];
3     for(int i = 0; i < a.size(); ++i) len[i] = 1;
4     int maxi = -1;
5     for(int i = 0; i < a.size(); ++i)
6         for(int j = i + 1; j < a.size(); ++j)
7             if(a[i] <= a[j]) len[j] = max(len[j],
                    len[i] + 1);
8
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 <<
                    "," << strB[j-1] << endl;
10            j--;
11        }
12        return;
13    }
14    if(strA[i-1] == strB[j-1]){
15        backtracking(i-1, j-1);
16    }
17    else{
18        if(dis[i][j] == dis[i-1][j-1] + 1){
19            cout << cnt++ << " Replace " << i << ","
                    << strB[j-1] << endl;
20            backtracking(i-1, j-1);
21        }
22        else if(dis[i][j] == dis[i-1][j] + 1){
23            cout << cnt++ << " Delete " << i << endl;
24            backtracking(i-1, j);
25        }
26        else if(dis[i][j] == dis[i][j-1] + 1){
27            cout << cnt++ << " Insert " << i + 1 <<
                    "," << strB[j-1] << endl;
28            backtracking(i, j-1);
29        }
30    }
31 }
32 void MED(){
33     // 由於 B 是 0，所以 A 轉換成 B
            時每個字元都要被刪除
34     for(int i = 0; i <= strA.size(); ++i) dis[i][0] =
            i;
35     // 由於 A 是 0，所以 A 轉換成 B
            時每個字元都需要插入
```

```
36      for(int j = 0; j <= strB.size(); ++j) dis[0][j] =
            j;
37      for(int i = 1; i <= strA.size(); ++i){
38          for(int j = 1; j <= strB.size(); ++j){
39              // 字元相同代表不需修改，修改距離直接延續
40              if(strA[i-1] == strB[j-1]) dis[i][j] =
                    dis[i-1][j-1];
41              else{
42                  // 取 replace , delete , insert
                        最小，選其 +1 為最少編輯距離
43                  dis[i][j] = min(dis[i-1][j-1],
                        min(dis[i-1][j], dis[i][j-1])) +
                        1;
44              }
45          }
46      }
47 }
```

5 Enumerate

5.1 Halfcut Enumerate

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

6 Graph

6.1 SPFA

```
1 bool SPFA(int s){
2     // 記得初始化這些陣列
3     int cnt[1000+5], dis[1000+5];
4     bool inqueue[1000+5];
5     queue<int> q;
6
7     q.push(s);
8     dis[s] = 0;
9     inqueue[s] = true;
10    cnt[s] = 1;
11    while(!q.empty()){
12        int now = q.front();
13        q.pop();
14        inqueue[now] = false;
15
16        for(auto &e : G[now]){
```

```
17            if(dis[e.t] > dis[now] + e.w){
18                dis[e.t] = dis[now] + e.w;
19                if(!inqueue[e.t]){
20                    cnt[e.t]++;
21                    if(cnt[e.t] > m){
22                        return false;
23                    }
24                    inqueue[e.t] = true;
25                    q.push(e.t);
26                }
27            }
28        }
29    }
30    return true;
31 }
```

6.2 Dijkstra

```
1 /* Dijkstra 最短路徑 */
2 struct Edge{
3     int v, w;
4 };
5 struct Item{
6     int u, dis;
7     // 取路徑最短
8     bool operator < (const Item &other) const{
9         return dis > other.dis;
10    }
11 };
12 int dis[maxn];
13 vector<Edge> G[maxn];
14 void dijkstra(int s){
15     for(int i = 0; i <= m; i++){
16         dis[i] = inf;
17     }
18     dis[s] = 0;
19     priority_queue<Item> pq;
20     pq.push({s, 0});
21     while(!pq.empty()){
22         // 取路徑最短的點
23         Item now = pq.top();
24         pq.pop();
25         if(now.dis > dis[now.u]){
26             continue;
27         }
28         // 把與 now.u 相連的點都跑一遍
29         for(Edge e : G[now.u]){
30             if(dis[e.v] > now.dis + e.w){
31                 dis[e.v] = now.dis + e.w;
32                 pq.push({e.v, dis[e.v]});
33             }
34         }
35     }
36 }
```

6.3 Floyd Warshall

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

```
14        }
15 }
```

6.4 Disjoint set Kruskal

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

6.5 Bipatirate

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

6.6 Hungarian algorithm

```
 1 /* 匈牙利演算法 */
 2 const int maxn = 500+5;
 3 int t, N, bn, gn, match[maxn];
 4 bool visited[maxn];
 5 vector<vector<int>> G(maxn);
 6 struct People{
 7     int h;
 8     string music, sport;
 9     People(){}
10     People(int h, string music, string sport){
11         this->h = h;
12         this->music = music;
13         this->sport = sport;
14     }
15 }lef[maxn], rig[maxn];
16 bool check(People boy, People girl){
17     if(abs(boy.h - girl.h) <= 40 && boy.music ==
           girl.music && boy.sport != girl.sport) return
           true;
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,
                    music, sport);
52              else rig[gn++] = People(h, music, sport);
53          }
54          for(int i = 0; i < bn; ++i){
55              for(int j = 0; j < gn; ++j)
56                  if(check(lef[i], rig[j]))
                        G[i].emplace_back(j);
57          }
58          cout << N - Hungarian() << endl;
59      }
60  }
```

6.7 LCA

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

6.8 Trie

```
1  /* Trie 字典樹 */
2  struct Tire{
3      int path;
4      map<string, int> G[maxn];
5      void init(){
6          path = 1;
7          G[0].clear();
8      }
9      void insert(string str){
10         int u = 0;
```

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

7 Math

7.1 Hash

```
1  /* 建議搭配 Other - Stammering_Aliens 食用 */
2  #define ull unsigned long long int
3  const int maxn = 40000+5;
4  const ull seed = 131;
5  ull pw[maxn], hhash[maxn], hhash2[maxn];
6  char str[maxn];
7  void init(){
8      hhash[0] = 0;
9      for(int i = len-1; i >= 0; --i)
10         hhash[i] = (hhash[i+1] * seed + str[i]);
11 }
```

8 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
             idx 0 到結束)
19        p = strstr(s, ss);
20        cout << p << endl;
21        // strstr 也可以單純用來找字串
22        if(p != NULL) cout << "yes" << endl;
23        else cout << "no" << enld;
24    }
25 }
```

8.3   setprecision

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

8.4   GCD LCM

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

8.5   reverse

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

8.6   sort

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

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();
         iter_r++)
13        cout<<iter_r->first<<"
             "<<iter_r->second<<endl;
14
15    iter = mp.find("r123");
16    mp.erase(iter);
17
18    iter = mp.find("r123");
19    if(iter != mp.end())
20        cout<<"Find, the value is
             "<<iter->second<<endl;
21    else
22        cout<<"Do not Find"<<endl;
23
24    mp.clear();
25    mp.erase(mp.begin(), mp.end());
26 }
```

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
         st.end()
7     if (iter != st.end()) {
8         cout << "Found: " << *iter << endl;
9     } 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{
```

```cpp
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   // 求兩點的 LCA（利用倍增法）
33   int LCA(int a, int b){
34       if (depth[b] < depth[a]) swap(a, b);
35       if (depth[a] != depth[b]){
36           int dif = depth[b] - depth[a];
37           for (int i = 0; i < 20; i++){
38               if (dif & 1) b = parent[b][i];
39               dif >>= 1;
40           }
41       }
42       if (a == b) return a;
43       for (int i = 19; i >= 0; i--){
44           if (parent[a][i] != parent[b][i]){
45               a = parent[a][i];
46               b = parent[b][i];
47           }
48       }
49       return parent[a][0];
50   }
51   long long int dist[maxn];
52   // 從 0 開始到每個點的距離
53   void distance(){
54       for (int u = 0; u < N; ++u){
55           for(int i = 0; i < G[u].size(); ++i){
56               dist[G[u][i].v] = dist[u] + G[u][i].w;
57   }
58   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;
```

```cpp
80           int lca = LCA(a, b);
81           if(space) cout << " ";
82           space = true;
83           cout << (dist[a] + dist[b]) - (dist[lca]
                     * 2);
84           }
85           cout << endl;
86       }
87   }
```

9.2 Binary codes

```cpp
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   }
16   int main(){
17       cin >> n;
18       for(int i = 0; i < n; ++i){
19           cin >> back[i];
20       zero = 0;
21       BWT();
22   }
```

9.3 Disk Tree

```cpp
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();
                 ++i){
27               for(int j = 0; j < space; ++j)
28                   cout << " ";
29               cout << i->first << endl;
30               put(i->second, space + 1);
31           }
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 }
```

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

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
       時可以放入的最大 value
4  0 0 0 30 30 30 30 30 30
5  w[i]: 4
6  0 0 0 30 50 50 50 80 80
7  w[i]: 5
8  0 0 0 30 50 60 60 80 90  */
9  int main(){
10     int N, W;
11     cin >> N >> W;
12     int w[100000+5], v[100000+5];
13     for(int i = 0; i < N; i++)
14         cin >> w[i] >> v[i];
15     long long int dp[100000+5];
16     memset(dp, 0, sizeof(dp));
17     for(int i = 0; i < N; i++)
18         for(int j = W; j >= w[i]; j--)
19             dp[j] = max(dp[j], dp[j - w[i]] + v[i]);
20     cout << dp[W] << endl;
21 }
```

11 Math

11.1 Big Mod

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

11.2 How Many 0s

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

11.3 ORXOR

```
1  /* bitwise operator 二進位制數論
2  如何切區段，之所以要1<<n是為了可以跑000~111
3  i = 0，binary i = 000
4  0 : 1 5 7
5  i = 1，binary i = 001
6  1 : 1 5 7
7  i = 2，binary i = 010，看得出來切了一刀
8  2 : 1 | 5 7
9  i = 3，binary i = 011
10 3 : 1 | 5 7
11 i = 4，binary i = 100，為了要切在index=2，所以才要1<<j
12 4 : 1 5 | 7
```

```
13  i = 5，binary i = 101
14  5 : 1 5 | 7
15  i = 6，binary i = 110
16  6 : 1 | 5 | 7
17  i = 7，binary i = 111
18  7 : 1 | 5 | 7
19  可以觀察出來，前兩位 bit 是 1 時代表的意義是切在哪裡
        */
20  int main(){
21      int n; cin >> n;
22      int num[20+7];
23      memset(num, 0, sizeof(num));
24      for(int i = 1; i <= n; i++)
25          cin >> num[i];
26      // 不知道為甚麼只有 2147483647 給過
27      int mini = 2147483647;
28      // 1 << n = n * 2
29      for(int i = 0; i < (1 << n); i++){
30          int XOR = 0, OR = 0;
31          for(int j = 1; j <= n; j++){
32              OR |= num[j];
33              if((i & (1 << j))){
34                  XOR ^= OR;
35                  OR = 0;
36              }
37          }
38          XOR ^= OR;
39          mini = min(mini, XOR);
40      }
41      cout << mini << endl;
42  }
```

12 Segement Tree

12.1 Frequent values

```
1   /* Segement Tree & RMQ (Range Sum Query)
2   idx:  1   2   3   4   5   6   7   8   9   10
3   num: -1  -1   1   1   1   1   3  10  10  10
4   fre:  2   2   4   4   4   4   1   3   3   3
5   border
6   left: 1   1   3   3   3   3   7   8   8   8
7   right:2   2   6   6   6   6   7  10  10  10 */
8   # define Lson(x) x << 1
9   # define Rson(x) (x << 1) + 1
10  const int maxn = 1e5+5;
11  struct Tree{
12      int lef, rig, value;
13  }tree[4 * maxn];
14  struct Num{
15      int lef, rig, value, fre;
16  }num[maxn];
17  // 建立 segement tree
18  void build(int lef, int rig, int x){
19      tree[x].lef = lef;
20      tree[x].rig = rig;
21      // 區塊有多長，題目詢問的重點
22      if(lef == rig){
23          tree[x].value = num[lef].fre;
24          return;
25      }
26      int mid = (lef + rig) >> 1;
27      build(lef, mid, Lson(x));
28      build(mid + 1, rig, Rson(x));
29      tree[x].value = max(tree[Lson(x)].value,
              tree[Rson(x)].value);
30  }
31  // 查詢 segement tree
32  int query(int lef, int rig, int x){
33      // 題目所查詢的區間剛好在同個區塊上，num[lef].v
              == num[rig].v
34      if(num[lef].value == num[rig].value) return rig -
              lef + 1;
35      int ans = 0;
```

```
36      // 查詢的左區間邊界切到區塊，且此區間有數個區塊
37      if(lef > num[lef].lef){
38          // 計算切到的區間大小
39          ans = num[lef].rig - lef + 1;
40          //
              更新左邊界至被切區塊的右邊界加一，就不會切到區塊
41          lef = num[lef].rig + 1;
42      }
43      // 查詢的右區間邊界切到區塊，且此區間有數個區塊
44      if(rig < num[rig].rig){
45          // 計算切到的區間大小，並找出最大
46          ans = max(ans, rig - num[rig].lef + 1);
47          // 更新右邊界
48          rig = num[rig].lef - 1;
49      }
50      //
              如果左邊界大於右邊界，表示不需要再進行查詢直接回傳答案
51      if(lef > rig) return ans;
52      if(tree[x].lef >= lef && tree[x].rig <= rig)
              return tree[x].value;
53      int mid = (tree[x].lef + tree[x].rig) >> 1;
54      if(lef <= mid) ans = max(ans, query(lef, rig,
              Lson(x)));
55      if(mid < rig) ans = max(ans, query(lef, rig,
              Rson(x)));
56      return ans;
57  }
58  int main(){
59      int n, q;
60      while(cin >> n && n){
61          cin >> q;
62          int start = 1;
63          for(int i = 1; i <= n; ++i){
64              cin >> num[i].value;
65              if(num[i].value != num[i-1].value){
66                  for(int j = start; j < i; ++j){
67                      num[j].rig = i - 1;
68                      num[j].fre = i - start;
69                  }
70                  start = num[i].lef = i;
71              }
72              else num[i].lef = start;
73          }
74          // 最後一段 [start, n]
75          for(int j = start; j <= n; ++j){
76              num[j].rig = n;
77              num[j].fre = n - start + 1;
78          }
79          build(1, n, 1);
80          int lef, rig;
81          for(int i = 0; i < q; ++i){
82              cin >> lef >> rig;
83              cout << query(lef, rig, 1) << endl;
84          }
85      }
86  }
```

13 Dijkstra

13.1 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
                       紀錄從終點到每個點的距離
63         memset(dp, -1, sizeof(dp));
64         for(int i = 1; i <= n; ++i){
65             path[i].clear();
66             for(int j = 0; j < G[i].size(); ++j){
67                 int v = G[i][j].v;
68                 // 如果到 v 的距離比到 i
                          遠，代表從起點經過 i 再到 v
69                 if(dis[i] > dis[v])
70                     path[i].push_back(v);
71             }
72         }
73         cout << dfs(1) << endl;
74     }
75 }
```

14 Kruskal

14.1 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] =
           find(parent[x]);
23 }
24 bool unite(int a, int b){
25     int x = find(a);
26     int y = find(b);
27     if(x == y) return false;
28     parent[x] = y;
29     return true;
30 }
31 double kruskal(){
32     m = 0; // m: 邊的數量
33     for(int i = 0; i < n; ++i)
34         for(int j = i + 1; j < n; ++j)
35             edge[m++] = (Edge){i, j, dist(i, j)};
36     sort(edge, edge + m);
37     for(int i = 0; i < n; ++i){
38         parent[i] = i;
39         G[i].clear();
40     }
41     double total = 0.0;
42     int edge_cnt = 0;
43     for(int i = 0; i < m; ++i){
44         int u = edge[i].u, v = edge[i].v;
45         double cnt = edge[i].w;
46         if(unite(u, v)){
47             G[u].push_back((Edge){u, v, cnt});
48             G[v].push_back((Edge){v, u, cnt});
49             total += cnt;
50             if(++edge_cnt == n-1) break;
51         }
52     }
53     return total;
54 }
55 double maxcost[maxn][maxn];
56 bool visited[maxn];
57 void dfs(int u){
58     visited[u] = true;
59     for(int i = 0; i < G[u].size(); ++i){
60         int v = G[u][i].v;
61         if(visited[v]) continue;
62         double cost = G[u][i].w;
63         maxcost[u][v] = maxcost[v][u] = cost;
64         // 更新 MST 樹上的點到 v 點的距離
65         for(int j = 0; j < n; ++j)
66             if(visited[j])
67                 maxcost[j][v] = maxcost[v][j] =
                       max(maxcost[j][u], cost);
68         dfs(v);
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 -
                    maxcost[i][j]));
81      printf("%.2lf\n", ans);
82  }
83  int main(){
84      int t;
85      scanf("%d", &t);
86      while(t--){
87          scanf("%d", &n);
88          for(int i = 0; i < n; ++i)
89              scanf("%d%d%d", &x[i], &y[i], &p[i]);
90          solve();
91      }
92      return 0;
93  }
```

15 Bipartite Graph

15.1 SAM I AM

```cpp
/* 二分圖匹配 + 最小點覆蓋 */
const int maxn = 1000+5;
int R, C, N;
bool arr[maxn][maxn], visitX[maxn], visitY[maxn];
int matchX[maxn], matchY[maxn];
int dfs(int x){
    visitX[x] = true;
    for(int y = 1; y <= C; ++y){
        if(arr[x][y] && !visitY[y]){
            visitY[y] = true;
            if(matchY[y] == 0 || dfs(matchY[y])){
                matchX[x] = y;
                matchY[y] = x;
                return 1;
            }
        }
    }
    return 0;
}
int Match(){
    int sum = 0;
    memset(matchX, 0, sizeof(matchX));
    memset(matchY, 0, sizeof(matchY));
    for(int i = 1; i <= R; ++i){
        memset(visitX, false, sizeof(visitX));
        memset(visitY, false, sizeof(visitY));
        sum += dfs(i);
    }
    return sum;
}
int main(){
    while(cin >> R >> C >> N && R && C && N){
        memset(arr, false, sizeof(arr));
        memset(visitX, false, sizeof(visitX));
        memset(visitY, false, sizeof(visitY));
        int row, col;
        for(int i = 0; i < N; ++i){
            cin >> row >> col;
            arr[row][col] = true;
        }
        int cnt = Match();
        cout << cnt;
        memset(visitX, 0, sizeof(visitX));
        memset(visitY, 0, sizeof(visitY));
        for(int i = 1; i <= R; ++i){
            if(matchX[i] == 0) dfs(i);
        for(int i = 1; i <= R; ++i)
            if(!visitX[i]) cout << " r" << i;
        for(int i = 1; i <= C; ++i)
            if(visitY[i]) cout << " c" << i;
        cout << endl;
    }
}
```