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
#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 Trie

```cpp
const int MAXL = ; // 自己填
const int MAXC = ;
struct Trie {
  int nex[MAXL][MAXC];
  int len[MAXL];
  int sz;
  void init() {
    memset(nex, 0, sizeof(nex));
    memset(len, 0, sizeof(len));
    sz = 0;
  }
  void insert(const string &str) {
    int p = 0;
    for (char c : str) {
      int id = c - 'a';
      if (!nex[p][id]) {
        nex[p][id] = ++sz;
      }
      p = nex[p][id];
    }
    len[p] = str.length();
  }
```

```
23    vector<int> find(const string &str, int i) {
24        int p = 0;
25        vector<int> ans;
26        for (; i < str.length(); i++) {
27            int id = str[i] - 'a';
28            if (!nex[p][id]) {
29                return ans;
30            }
31            p = nex[p][id];
32            if (len[p]) {
33                ans.pb(len[p]);
34            }
35        }
36        return ans;
37    }
38 };
```

2.4 BWT

```
1  /*BWT 資料轉換演算法*/
2  void BWT(){
3      for(int i = 0; i < n; ++i){
4          if(back[i] == 0)
5              mini[zero++] = i;
6      for(int i = 0; i < n; ++i)
7          if(back[i] == 1)
8              mini[zero++] = i;
9      int ptr = mini[0];
10     for(int i = 0; i < n; ++i){
11         cout << back[ptr] << " ";
12         ptr = mini[ptr];
13     }
14     cout << endl;
15 }
```

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 }
```

```
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 Josephus

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

4.3 LCS

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

4.4 LIS

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

5   Enumerate

5.1   Halfcut Enumerate

```cpp
/* 折半枚舉 */
void dfs(set<long long int> &s, int depth, int T,
    long long int sum){
    if(depth >= T){
        s.insert(sum);
        return;
    }
    dfs(s, depth + 1, T, sum); // 取或不取的概念
    dfs(s, depth + 1, T, sum + A[depth]);
}
int main(){
    int N, T;
    set<long long int> s1, s2;
    cin >> N >> T;
    for(int i = 0; i < N; ++i) cin >> A[i];
    dfs(s1, 0, N/2, 0); // 折半枚舉
    dfs(s2, N/2, N, 0);
    long long int ans = 0;
    // 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
    for(auto &x : s1){
        auto it = s2.upper_bound(T - x);
        long long int y = *(--it);
        if(x + y <= T) ans = max(ans, x + y);
    }
    cout << ans << endl;
}
```

6   Graph

6.1   SPFA

```cpp
bool SPFA(int s){
    // 記得初始化這些陣列
    int cnt[1000+5], dis[1000+5];
    bool inqueue[1000+5];
    queue<int> q;

    q.push(s);
    dis[s] = 0;
    inqueue[s] = true;
    cnt[s] = 1;
    while(!q.empty()){
        int now = q.front();
        q.pop();
        inqueue[now] = false;

        for(auto &e : G[now]){
            if(dis[e.t] > dis[now] + e.w){
                dis[e.t] = dis[now] + e.w;
                if(!inqueue[e.t]){
                    cnt[e.t]++;
                    if(cnt[e.t] > m){
                        return false;
                    }
                    inqueue[e.t] = true;
                    q.push(e.t);
                }
            }
        }
    }
    return true;
}
```

6.2   Dijkstra

```cpp
struct Item{
    int u, dis;
    // 取路徑最短
    bool operator < (const Item &other) const{
        return dis > other.dis;
    }
};
int dis[maxn];
vector<Edge> G[maxn];
void dijkstra(int s){
    for(int i = 0; i <= n; i++){
        dis[i] = inf;
    }
    dis[s] = 0;
    priority_queue<Item> pq;
    pq.push({s, 0});
    while(!pq.empty()){
        // 取路徑最短的點
        Item now = pq.top();
        pq.pop();
        if(now.dis > dis[now.u]){
            continue;
        }
        // 鬆弛更新，把與 now.u 相連的點都跑一遍
        for(Edge e : G[now.u]){
            if(dis[e.v] > now.dis + e.w){
                dis[e.v] = now.dis + e.w;
                pq.push({e.v, dis[e.v]});
            }
        }
    }
}
```

6.3   Floyd Warshall

```cpp
void floyd_warshall(){
    for(int i = 0; i < n; i++){
        for(int j = 0; j < n; j++){
            G[i][j] = INF;
        }
        G[i][i] = 0;
    }
    for (int k = 0; k < n; k++){      //
        嘗試每一個中繼點
        for (int i = 0; i < n; i++){ //
            計算每一個i點與每一個j點
            for (int j = 0; j < n; j++){
                G[i][j] = min(G[i][j], G[i][k] +
                    G[k][j]);
            }
        }
    }
}
```

6.4   Disjoint set Kruskal

```cpp
struct Edge{
    int u, v, w;
    // 用權重排序 由大到小
    bool operator < (const Edge &other) const{
        return w > other.w;
    }
}edge[maxn];
// disjoint set
int find(int x){
    if(parent[x] < 0){
        return x;
    }
    else{
        return parent[x] = find(parent[x]);
    }
}
void unite(int a, int b){
```

```
18    a = find(a);
19    b = find(b);
20
21    if(a != b){
22      if(parent[a] < parent[b]){
23        parent[a] += parent[b];
24        parent[b] = a;
25      }
26      else{
27        parent[b] += parent[a];
28        parent[a] = b;
29      }
30    }
31 }
32 void kruskal(){
33    memset(parent, -1, sizeof(parent));
34    sort(edge, edge + m);
35    int i, j;
36    for(i = 0, j = 0; i < n - 1 && j < m; i++){
37        // 如果 u 和 v 的祖先相同，則 j++
                   (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
                   (連接兩顆子生成樹)
40        unite(edge[j].u, edge[j].v);
41        j++;
42    }
43 }
```

6.5　KM

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

```
40                 int z = my[y];
41                 *qb++ = z; p[z] = x; vx[z] = true;
                           relax(z);
42             }
43     return false;
44 }
45 bool branch2(){ // 延展交錯樹：使用新添的等邊
46     for (int y=0; y<Y; ++y){
47         if (!vy[y] && dy[y] == 0){
48             vy[y] = true;
49             if (my[y] == -1){
50                 augment(pdy[y], y);
51                 return true;
52             }
53             int z = my[y];
54             *qb++ = z; p[z] = pdy[y]; vx[z] = true;
                       relax(z);
55         }
56     }
57     return false;
58 }
59 int Hungarian(){
60     // 初始化vertex labeling
61     //  memset(lx, 0, sizeof(lx));  // 任意值皆可
62     memset(ly, 0, sizeof(ly));
63     for (int x=0; x<X; ++x)
64         for (int y=0; y<Y; ++y)
65             lx[x] = max(lx[x], adj[x][y]);
66
67     // X側每一個點，分別建立等邊交錯樹。
68     memset(mx, -1, sizeof(mx));
69     memset(my, -1, sizeof(my));
70     for (int x=0; x<X; ++x){
71         memset(vx, false, sizeof(vx));
72         memset(vy, false, sizeof(vy));
73         memset(dy, 0x7f, sizeof(dy));
74         qf = qb = q;
75         *qb++ = x; p[x] = -1; vx[x] = true; relax(x);
76         while (true){
77             if (branch1()) break;
78             reweight();
79             if (branch2()) break;
80         }
81     }
82     // 計算最大權完美匹配的權重
83     int weight = 0;
84     for (int x=0; x<X; ++x)
85         weight += adj[x][mx[x]];
86     return weight;
87 }
```

6.6　Dinic

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

```
23          for (int i = adj[a]; i != -1; i = e[i].next){
24              int b = e[i].b;
25              if (e[i].r > 0 && !visit[b]){
26                  d[b] = d[a] + 1;
27                  visit[b] = true;
28                  q[qn++] = b;
29                  if (b == t) return d[t];
30              }
31          }
32      }
33      return V;
34 }
35 int DFS(int a, int df, int s, int t){ //
        求出一條最短擴充路徑，並擴充流量
36      if (a == t) return df;
37      if (visit[a]) return 0;
38      visit[a] = true;
39      for (int i = adj[a]; i != -1; i = e[i].next){
40          int b = e[i].b;
41          if (e[i].r > 0 && d[a] + 1 == d[b]){
42              int f = DFS(b, min(df, e[i].r), s, t);
43              if (f){
44                  e[i].r -= f;
45                  e[i^1].r += f;
46                  return f;
47              }
48          }
49      }
50      return 0;
51 }
52 int dinitz(int s, int t){
53      int flow = 0;
54      while (BFS(s, t) < V)
55          while (true){
56              memset(visit, false, sizeof(visit));
57              int f = DFS(s, 1e9, s, t);
58              if (!f) break;
59              flow += f;
60          }
61      return flow;
62 }
```

6.7 Bipatirate

```
1  const int maxn = 300 + 5;
2  int n, color[maxn];
3  vector<vector<int>> v(maxn);
4  bool dfs(int s){
5      for(auto it : v[s]){
6          if(color[it] == -1){
7              color[it] = 3 - color[s];
8              if(!dfs(it)){
9                  return false;
10             }
11         }
12         if(color[s] == color[it]){
13             return false;
14         }
15     }
16     return true;
17 }
18 void isBipatirate(){
19     bool flag = true;
20     for(int i = 1; i <= n; ++i){
21         if(color[i] == -1){
22             color[i] = 1;
23             flag &= dfs(i);
24         }
25     }
26     if(flag){
27         cout << "YES" << endl;
28     }
29     else{
30         cout << "NO" << endl;
31     }
```

```
32 }
33 int main(){
34     while(cin >> n && n){
35         for(int i = 1; i <= n; ++i) v[i].clear();
36         memset(color, -1, sizeof(color));
37         int a, b;
38         while(cin >> a >> b && (a || b)){
39             v[a].emplace_back(b);
40             v[b].emplace_back(a);
41         }
42         isBipatirate();
43     }
44 }
```

6.8 Hungarian algorithm

```
1  const int maxn = 500+5;
2  int t, N, bn, gn, match[maxn];
3  bool visited[maxn];
4  vector<vector<int>> G(maxn);
5  struct People{
6      int h;
7      string music, sport;
8      People(){}
9      People(int h, string music, string sport){
10         this->h = h;
11         this->music = music;
12         this->sport = sport;
13     }
14 }lef[maxn], rig[maxn];
15 bool check(People boy, People girl){
16     if(abs(boy.h - girl.h) <= 40 && boy.music ==
            girl.music && boy.sport != girl.sport) return
            true;
17     return false;
18 }
19 bool dfs(int s){
20     for(int i = 0; i < G[s].size(); ++i){
21         int v = G[s][i];
22         if(visited[v]) continue;
23         visited[v] = true;
24         if(match[v] == -1 || dfs(match[v])){
25             match[v] = s;
26             return true;
27         }
28     }
29     return false;
30 }
31 int Hungarian(){
32     int cnt = 0;
33     memset(match, -1, sizeof(match));
34     for(int i = 0; i < bn; ++i){
35         memset(visited, false, sizeof(visited));
36         if(dfs(i)) cnt++;
37     }
38     return cnt;
39 }
40 int main(){
41     cin >> t;
42     while(t--){
43         cin >> N;
44         bn = 0, gn = 0;
45         for(int i = 0; i <= N; ++i) G[i].clear();
46         int h;
47         string sex, music, sport;
48         for(int i = 0; i < N; ++i){
49             cin >> h >> sex >> music >> sport;
50             if(sex == "M") lef[bn++] = People(h,
                    music, sport);
51             else rig[gn++] = People(h, music, sport);
52         }
53         for(int i = 0; i < bn; ++i){
54             for(int j = 0; j < gn; ++j)
55                 if(check(lef[i], rig[j]))
                        G[i].emplace_back(j);
56         }
```

```
57        cout << N - Hungarian() << endl;
58    }
59 }
```

6.9 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 }
```

7 Other

7.1 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 // 求兩點的 LCA（利用倍增法）
32 int LCA(int a, int b){
33     if (depth[b] < depth[a]) swap(a, b);
34     if (depth[a] != depth[b]){
35         int dif = depth[b] - depth[a];
36         for (int i = 0; i < 20; i++){
37             if (dif & 1) b = parent[b][i];
38             dif >>= 1;
39         }
40     }
41     if (a == b) return a;
42     for (int i = 19; i >= 0; i--){
43         if (parent[a][i] != parent[b][i]){
44             a = parent[a][i];
45             b = parent[b][i];
46         }
47     }
48     return parent[a][0];
49 }
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 int main(){
58     while(cin >> N && N){
59         memset(dist, 0, sizeof(dist));
60         memset(parent, 0, sizeof(parent));
61         memset(depth, 0, sizeof(depth));
62         memset(siz, 0, sizeof(siz));
63         for(int i = 0; i <= N; ++i){
64             G[i].clear();
65         }
66         for(int i = 1; i < N; ++i){
67             int u, w;
68             cin >> u >> w;
69             G[u].push_back({i, w});
70             parent[i][0] = u;
71         }
72         find_parent();
73         dfs(0, 0);
74         distance();
75         int s; cin >> s;
76         bool space = false;
77         for(int i = 0; i < s; ++i){
78             int a, b;
79             cin >> a >> b;
80             int lca = LCA(a, b);
81             if(space) cout << " ";
82             space = true;
83             cout << (dist[a] + dist[b]) - (dist[lca]
                    * 2);
84         }
85         cout << endl;
86     }
87 }
```

7.2 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      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  }
```

8 DP

8.1 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  }
```

8.2 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  }
```

8.3 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
        -1 7 -1 -1 8 -1 -1 9 -1 -1 10 -1 -1 11 -1 -1 12
        -1 -1 13 -1 -1 14 -1 -1 15 -1 -1 16 -1 -1 17 -1
        -1 18
 5  吃 5 分鐘漢堡時 (更新)
 6  0 -1 -1 1 -1 -1 1 2 -1 2 3 2 3 4 3 4 5 4 5 6 5 6 7 6 7 8
        7 8 9 8 9 10 9 10 11 10 11 12 11 12 13 12 13 14
        13 14 15 14 15 16 15 16 17 16 17 18
 7  只有當該時間可剛好吃滿漢堡時會更新
 8  全部初始設 -1，用以判斷 譬如當 1 分鐘時
        吃不了任何漢堡*/
 9  int main(){
10      int m, n, t;
11      while(cin >> m >> n >> t){
12          int dp[10000+5];
13          memset(dp, -1, sizeof(dp));
14          dp[0] = 0;
15          for(int i = m; i <= t; i++)
16              if(dp[i - m] != -1)
17                  dp[i] = max(dp[i], dp[i - m] + 1);
18          for(int i = n; i <= t; i++)
19              if(dp[i - n] != -1)
20                  dp[i] = max(dp[i], dp[i - n] + 1);
21          // 時間無法剛好吃滿的時候
22          if(dp[t] == -1){
23              for(int i = t; i >= 0; i--)
24                  if(dp[i] != -1){
25                      cout << dp[i] << " " << t - i <<
                            endl;
26                      break;
27                  }
28          }
29          else cout << dp[t] << endl;
30      }
31  }
```

8.4 Let Me Count The Ways

```
 1  /* dp - 時間/數量 - 硬幣排序
 2  要湊出 17
 3  1 1 1 1 1 2 2 2 2 2 4 4 4 4 4 6 6 */
 4  int main(){
 5      long long int n;
 6      long long int dp[30000+5];
 7      int coin[] = {1, 5, 10, 25, 50};
 8      memset(dp, 0, sizeof(dp));
 9      // 直接把 dp 做好
10      dp[0] = 1;
11      for(int i = 0; i < 5; i++)
12          for(int j = coin[i]; j < 30000+5; j++)
13              if(dp[j - coin[i]] != -1)
14                  dp[j] += dp[j - coin[i]];
15      while(cin >> n){
16          if(dp[n] == 1)
17              cout << "There is only " << dp[n] << "
                    way to produce " << n << " cents
                    change." << endl;
18          else
19              cout << "There are " << dp[n] << " ways
                    to produce " << n << " cents change."
                    << endl;
20      }
21  }
```

8.5 Luggage

```cpp
/* dp 背包 - 重量/是否成立
7 7 13 1
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
    1
Note: dp[0] = true */
int main(){
    int t;
    cin >> t;
    cin.ignore();
    while(t--){
        string str;
        getline(cin , str);
        vector<int> v;
        stringstream ss;
        int num, cnt = 0, sum = 0;;
        bool dp[4000+5];
        memset(dp, false, sizeof(dp));
        ss << str;
        while(ss >> num){
            cnt++;
            sum += num;
            v.emplace_back(num);
        }
        if(sum & 1){
            cout << "NO" << endl;
            continue;
        }
        dp[0] = true;
        for(int i = 0; i < v.size(); i++)
            for(int j = sum; j >= v[i]; j--)
                if(dp[j - v[i]])
                    dp[j] = true;
        cout << (dp[sum/2] ? "YES" : "NO") << endl;
    }
}
```

8.6 Partitioning by Palindromes

```cpp
/* string & dp - 字串長度判斷迴文
r a c e c a r
i = 0, j = 0
-> r = r ，dp[1] = dp[0] + 1 = 1
i = 1, j = 0
-> 因 a != r，dp[2] = 0x3f3f3f3f
i = 1, j = 1
-> 因 a = a, dp[2] = dp[1] + 1 = 2 */
bool check_palindromes(int lef, int rig){
    // 比較字串兩端都是迴文
    while(lef < rig){
        if(str[lef] != str[rig]) return 0;
        lef++;
        rig--;
    }
    return 1;
}
int main(){
    int t;
    cin >> t;
    while(t--){
        cin >> str;
        memset(dp, 0x3f3f3f3f, sizeof(dp));
        dp[0] = 0;
        for(int i = 0; i < str.size(); ++i)
            for(int j = 0; j <= i; ++j)
                if(str[i] == str[j])
                    if(check_palindromes(j, i))
                        if(dp[i+1] > dp[j] + 1)
                            dp[i+1] = dp[j] + 1;
        cout << dp[str.size()] << endl;
    }
}
```

8.7 SuperSale

```cpp
/* dp 背包 - 重量/價值/不可重複使用
第一個人的負重: 23
0 0 0 0 52 52 52 52 52 54 54 54 54 106 106 106 106
    106 106 106 106 106 151 151
第二個人的負重: 20
0 0 0 0 52 52 52 52 52 54 54 54 54 106 106 106 106
    106 106 106 106
第三個人的負重: 20
0 0 0 0 52 52 52 52 52 54 54 54 54 106 106 106 106
    106 106 106 106
第四個人的負重: 26
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 */
struct Edge{
    int p;
    int w;
}edge[1000+5];
int main(){
    int t;
    cin >> t;
    while(t--){
        int n; cin >> n;
        for(int i = 0; i < n; i++)
            cin >> edge[i].p >> edge[i].w;
        int g, total = 0;
        cin >> g;
        for(int i = 0; i < g; i++){
            int pw; in >> pw;
            int dp[30+5];
            memset(dp, 0, sizeof(dp));
            for(int j = 0; j < n; j++)
                for(int k = pw; k >= edge[j].w; k--)
                    dp[k] = max(dp[k], dp[k -
                        edge[j].w] + edge[j].p);
            total += dp[pw];
        }
        cout << total << endl;
    }
}
```

8.8 Walking on the Safe Side

```cpp
/* dp - 地圖更新
更新地圖
一張如下的地圖 其 dp 更新方法為加上和加左的路
0 0 0 0 0
0 1 0 0 0
0 0 0 1 0 1
0 0 0 0 0
1 1 1 1 1
1 0 1 2 3
1 1 0 2 0
1 2 2 4 4 */
bool mp[100+5][100+5];
long long int dp[100+5][100+5];
int main(){
    int t; cin >> t;
    bool space = false;
    while(t--){
        if(space) cout << endl;
        else space = true;
        int r, c; cin >> r >> c;
        cin.ignore();
        memset(mp, false, sizeof(mp));
        memset(dp, 0, sizeof(dp));
        string str;
        for(int i = 0; i < r; i++){
            getline(cin, str);
            int n, num;
            stringstream ss(str);
            ss >> n;
            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 }
```

8.9 Cutting Sticks

```
1  /* dp - 動態切割取最小
2  100
3  3
4  25 50 75
5  dp:
6  0 0 50 125 200
7  0 0 0 50 125
8  0 0 0 0 50
9  0 0 0 0 0
10 0 0 0 0 0 */
11 int main(){
12     int l;
13     while(cin >> l && l){
14         int n;
15         cin >> n;
16         vector<int> s(n+2);
17         s[0] = 0;
18         for(int i = 1; i <= n; ++i)
19             cin >> s[i];
20         // 從現在開始 n 的數量變為 n + 1
21         s[++n] = l;
22         int dp[n+5][n+5];
23         memset(dp, 0, sizeof(dp));
24         // r: 切幾段 b: 起點 c: 中間點 e: 終點
25         for(int r = 2; r <= n; ++r){
26             for(int b = 0; b < n; ++b){
27                 // 如果從 b 開始切 r 刀會超出長度就
                        break
28                 if(b + r > n) break;
29                 // e: 從 b 開始切 r 刀
30                 int e = b + r;
31                 dp[b][e] = 0x3f3f3f3f;
32                 // c: 遍歷所有從 b 開始到 e
                        結束的中間點
33                 for(int c = b + 1; c < e; ++c){
34                     // dp[b][c] 從 b 到 c 最少 cost +
                            dp[c][e] 從 c 到 e 最少 cost
35                     // s[e] - s[b] 兩段之間的 cost
36                     dp[b][e] = min(dp[b][e], dp[b][c]
                            + dp[c][e] + s[e] - s[b]);
37                 }
38             }
39         }
40         cout << "The minimum cutting is " << dp[0][n]
                << "." << endl;
41     }
42 }
```

8.10 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;
           i++){
25         total++;
26         if(n % pri[i]) continue;
27         prime++;
28         dp[n] += dfs(n/pri[i]);
29     }
30     // 算期望值
31     dp[n] = (dp[n] + total)/prime;
32     return dp[n];
33 }
34 int main(){
35     int t, num, ca = 1;
36     for(int i = 0; i <= N; i++)
37         dp[i] = -1;
38     Linear_Sieve();
39     cin >> t;
40     while(t--){
41         cin >> num;
42         cout << "Case " << ca++ << ": " << fixed <<
               setprecision(10) << dfs(num) << endl;
43     }
44 }
```

9 Math

9.1 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
```

9.2 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
            代表最後答案會是整數，否則會是分數
15     if((n * (n - 1)) % 4){
16         cout << ( (n * (n - 1)) / 2 ) << "/2"<< endl;
17     }
18     else{
19         cout << ( (n * (n - 1)) / 2 ) / 2 << endl;
20     }
21 }
```

9.3 Fraction Floor Sum

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

9.4 How Many 0s

```
1  /* 數論 */
2  int main(){
3      long long int n, m;
4      while(cin >> n >> m && (n >= 0) && (m >= 0)){
5          long long int total1 = 0, total2 = 0;
6          long long int ten = 1, tmp = n-1;
7          while(tmp >= 10){
8              if(tmp % 10 == 0){
9                  tmp /= 10;
10                 total1 += (tmp - 1) * ten + ((n-1) %
                       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 }
```

9.5 Number of Pairs

```
1  /* 數論
2  uper_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,
                   v.end(), r - v[i]) -
                   lower_bound(v.begin() + i + 1,
                   v.end(), l - v[i]));
24         cout << ans << endl;
25     }
26 }
```

9.6 ORXOR

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

```
37            }
38            XOR ^= OR;
39            mini = min(mini, XOR);
40        }
41        cout << mini << endl;
42 }
```

9.7 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 }
```

10 Binary Search

10.1 Fill the Containers

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

10.2 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(),
                   marble) - v.begin();
14             int rig = upper_bound(v.begin(), v.end(),
                   marble) - v.begin();
15             if(lef == rig) cout << marble << " not
                   found" << endl;
16             else{
17                 cout << marble << " found at " << lef
                       + 1 << endl;
18             }
19         }
20     }
21 }
```

11 Segement Tree

11.1 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
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
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 }
```

12 Bipartite Graph

12.1 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              //
                    如果與點相連又還未填色，填塞成與原點不同的
10             color[it] = 3 - color[s];
11             // 同樣對此點去判定與此點相連的點的填色
12             if(!dfs(it)) return false;
13         }
14         if(color[s] == color[it]){
```

```
15             // 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
                   1，並對與此點相連的點都 dfs 判定填色
26             color[i] = 1;
27             flag &= dfs(i);
28         }
29     }
30     if(flag) cout << "YES" << endl;
31     else cout << "NO" << endl;
32 }
33 int main(){
34     while(cin >> n && n){
35         for(int i = 1; i <= n; ++i) v[i].clear();
36         memset(color, -1, sizeof(color));
37         int a, b;
38         while(cin >> a >> b && (a || b)){
39             v[a].emplace_back(b);
40             v[b].emplace_back(a);
41         }
42         isBipatirate();
43     }
44 }
```

12.2 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 ==
           girl.music && boy.sport != girl.sport) return
           true;
21     return false;
22 }
23 bool dfs(int s){
24     for(int i = 0; i < G[s].size(); ++i){
25         int v = G[s][i];
26         if(visited[v]) continue;
27         visited[v] = true;
28         // 如果這個女生還沒被配對過，直接匹配
29         // 如果已經被配對，則根據這個女生所配對的對象
               dfs 重新匹配所有人的對象
30         if(match[v] == -1 || dfs(match[v])){
31             match[v] = s;
32             return true;
33         }
34     }
35     return false;
36 }
37 int Hungarian(){
38     int cnt = 0;
```

```
39    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 }
```

13　Function

13.1　strstr

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

13.2　substr

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

13.3　map set

```
1 .begin( ) // Return iterator to beginning
2 .end( ) // Return iterator to end
3 .empty( ) // 檢查是否為空
4 .size( )  // 回傳大小
```

```
5 mp.insert(pair<char,int>('a',100))
6 st.insert(100) // 插入 key、value
7 .erase( ) // 刪掉指定 key 和他的 value
8 .clear( ) // 清空整個 map
9 m.find( )
10 cout << "a => " << mymap.find('a')->second << endl;
11    // 找出 map 裡 key
       有沒有在裡面，如果有的話會回傳元素所在的 iterator，否則傳
12 s.count() // 返回某個值元素在 set 的個數
13 while( !mymap.empty()){
14    cout << mymap.begin()->first << " => " <<
          mymap.begin()->second << endl;
15    mymap.erase(mymap.begin());
16 }
17 for (auto it = mymap.begin(); it != mymap.end(); ++it)
18    cout << it->first << " => " << it->second << endl;
```

13.4　vector

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

13.5　setprecision

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

13.6　GCD LCM

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

13.7　reverse

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

13.8　CHAR

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

13.9 sort

```
priority_queue<int, vector<int>, less<int>> //大到小
priority_queue<int, vector<int>, greater<int>>
    //小到大

int arr[] = {4, 5, 8, 3, 7, 1, 2, 6, 10, 9};
    sort(arr, arr+10);

vector<int> v;
sort(v.begin(), v.end()); //小到大

int cmp(int a, int b){
    return a > b;
}
sort(v.begin(), v.end(), cmp); //大到小
```

13.10 struct

```
struct area{
    int a, b;
    bool operator<(const area rhs) const{
        return a > rhs.a || ( a == a && b > rhs.b);
    }
    bool operator!=(const area rhs) const{
        return a != rhs.a || b != rhs.b;
    }
};
```

13.11 deque

```
deque <int> que;
que.push_back(10);
que.push_front(20);
que.front()
que.back()
que.pop_front()
que.pop_back()
cout << "Element at position 2 : " << que.at(2) <<
    endl;
```

13.12 python template

```
import math
import operator

try:
    while(1):
        listx = []
        listx.append("...")
        list_s = sorted(listx) # 小到大
        list_s = sorted(listx, reverse = True) #
            大到小
        # max(listx)
        # min(listx)
        # sum(listx)
        # len(listx)
        dicty = {}
        dicty[key] = "value"
        dicty= sorted(dicty.items()) # by key
        dicty= sorted(dicty.items(),
            key=operator.itemgetter(1)) # by value
        # EOF寫法
        # 階層 math.factorial(3) == 6
        # 絕對值 math.fabs(x)
        # 無條件進位 math.ceil(3.1) == 3
        # 無條件捨去 math.floor(2.9) == 2
        # C n 取 k math.comb(n, k)
        # math.gcd
        # math.lcm
        # e 次 x 冪 math.exp(x)
except EOFError:
    pass
```