

Contents

1 Data Structure

1.1 Binary Search	1
1.2 BIT	1
1.3 Segment tree	1
1.4 Trie	1
1.5 BWT	2

2 Divide and Conquer

2.1 count inversions	2
----------------------	---

3 DP

3.1 Doubling	2
3.2 Josephus	2
3.3 LCS	2
3.4 LIS	2

4 Enumerate

4.1 Halfcut Enumerate	2
-----------------------	---

5 Graph

5.1 SPFA	3
5.2 Dijkstra	3
5.3 Floyd Warshall	3
5.4 Disjoint set Kruskal	3
5.5 KM	4
5.6 Dinic	4
5.7 Bipartite	5
5.8 Hungarian algorithm	5
5.9 LCA	5

6 Other

6.1 Ants Colony	6
6.2 Binary codes	6

7 Math

7.1 Big Mod	7
7.2 Bubble Sort Expect Value	7
7.3 Fraction Floor Sum	7
7.4 How Many Os	7
7.5 Number of Pairs	7
7.6 ORXOR	8
7.7 X drawing	8

8 Function

8.1 strstr	8
8.2 substr	8
8.3 map set	8
8.4 vector	8
8.5 setprecision	8
8.6 GCD LCM	8
8.7 reverse	9
8.8 CHAR	9
8.9 sort	9
8.10 struct	9
8.11 deque	9
8.12 python template	9

1 Data Structure

1.1 Binary Search

```

1 int binary_search(int arr[maxn], int lef, int rig,
2   int target){
3   if(lef > rig) return 0x3f3f3f3f;
4   int mid = (lef + rig) >> 1;
5   if(arr[mid] == target) return mid;
6   else if(arr[mid] > target){
7       return binary_search(arr, lef, mid - 1,
8         target);
9   }
10  else{
11      return binary_search(arr, mid + 1, rig,
12        target);
13  }
14 }
```

1.2 BIT

```

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

1.3 Segment tree

```

3 int dfs(int lef, int rig){
4     if(lef + 2 == rig){
5         if(num[lef] > num[rig-1]){
6             return lef;
7         }
8         else{
9             return rig-1;
10        }
11    }
12    int mid = (lef + rig)/2;
13    int p1 = dfs(lef, mid);
14    int p2 = dfs(mid, rig);
15    if(num[p1] > num[p2]){
16        return p1;
17    }
18    else{
19        return p2;
20    }
21 }
```

1.4 Trie

```

8 const int MAXL = ; // 自己填
9 const int MAXC = ;
10 struct Trie {
11     int nex[MAXL][MAXC];
12     int len[MAXL];
13     int sz;
14     void init() {
15         memset(nex, 0, sizeof(nex));
16         memset(len, 0, sizeof(len));
17         sz = 0;
18     }
19     void insert(const string &str) {
20         int p = 0;
21         for (char c : str) {
22             int id = c - 'a';
23             if (!nex[p][id]) {
24                 nex[p][id] = ++sz;
25             }
26             p = nex[p][id];
27         }
28         len[p] = str.length();
29     }
30     vector<int> find(const string &str, int i) {
31         int p = 0;
32         vector<int> ans;
33         for (; i < str.length(); i++) {
34             int id = str[i] - 'a';
35             if (!nex[p][id]) {
36                 return ans;
37             }
38             p = nex[p][id];
39             if (len[p]) {
40                 ans.pb(len[p]);
41             }
42         }
43     }
44 }
```

```

34     }
35     }
36     return ans;
37 }
38 };

```

1.5 BWT

```

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

```

2 Divide and Conquer

2.1 count inversions

```

1  /*逆序數對*/
2  int arr[maxn], buf[maxn];
3  int count_inversions(int lef, int rig){
4      if(rig - lef <= 1) return 0;
5      int mid = (lef + rig)/2;
6      int ans = count_inversions(lef, mid) +
7                count_inversions(mid, rig);
8      int i = lef, j = mid, k = lef;
9      while(i < mid || j < rig){
10         if(i >= mid) buf[k] = arr[j++];
11         else if(j >= rig) buf[k] = arr[i++];
12         else{
13             if(arr[i] <= arr[j]) buf[k] = arr[i++];
14             else{
15                 buf[k] = arr[j++];
16                 ans += mid - i;
17             }
18         }
19         k++;
20     }
21     for(int k = lef; k < rig; ++k) arr[k] = buf[k];
22     return ans;
23 }

```

3 DP

3.1 Doubling

```

1  /* 倍增 */
2  int LOG = sqrt(N); // 2^LOG >= N
3  vector<int> arr(N);
4  vector<vector<int>> dp(N, vector<int>(LOG));
5  for(int i = 0; i < N; ++i) cin >> arr[i];
6  int L, Q, a, b;
7  cin >> L >> Q;
8  for(int i = 0; i < N; ++i){
9      dp[i][0] = lower_bound(arr.begin(), arr.end(),
10                             arr[i] + L) - arr.begin();
11      if(dp[i][0] == N || arr[i] + L < arr[dp[i][0]])
12          dp[i][0] = -1;
13  }

```

```

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 }

```

3.2 Josephus

```

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

```

3.3 LCS

```

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

```

3.4 LIS

```

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

```

4 Enumerate

4.1 Halfcut Enumerate

```

1  /* 折半枚舉 */
2  void dfs(set<long long int> &s, int depth, int T,
3          long long int sum){

```

```

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
19     // 集合內小於等於 T-Sxi 中最大的數 Syj
20     for(auto &x : s1){
21         auto it = s2.upper_bound(T - x);
22         long long int y = *(--it);
23         if(x + y <= T) ans = max(ans, x + y);
24     }
25     cout << ans << endl;
26 }

```

5 Graph

5.1 SPFA

```

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

```

5.2 Dijkstra

```

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

```

```

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

```

5.3 Floyd Warshall

```

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

```

5.4 Disjoint set Kruskal

```

1 struct Edge{
2     int u, v, w;
3     // 用權重排序 由大到小
4     bool operator < (const Edge &other) const{
5         return w > other.w;
6     }
7 }edge[maxn];
8 // disjoint set
9 int find(int x){
10    if(parent[x] < 0){
11        return x;
12    }
13    else{
14        return parent[x] = find(parent[x]);
15    }
16 }
17 void unite(int a, int b){
18     a = find(a);
19     b = find(b);
20
21     if(a != b){
22         if(parent[a] < parent[b]){
23             parent[a] += parent[b];
24             parent[b] = a;
25         }
26     }
27 }

```

```

26     else{
27         parent[b] += parent[a];
28         parent[a] = b;
29     }
30 }
31 }
32 void kruskal(){
33     memset(parent, -1, sizeof(parent));
34     sort(edge, edge + m);
35     int i, j;
36     for(i = 0, j = 0; i < n - 1 && j < m; i++){
37         // 如果 u 和 v 的祖先相同, 則 j++
38         // (祖先相同代表會產生環 所以不要)
39         while(find(edge[j].u) == find(edge[j].v)) j++;
40         // 若都會產生環 則讓兩點之間產生橋
41         // (連接兩顆子生成樹)
42         unite(edge[j].u, edge[j].v);
43         j++;
44     }
45 }

```

5.5 KM

```

1  const int X = 50; // x的點數, 等於y的點數
2  const int Y = 50; // y的點數
3  int adj[X][Y]; // 精簡過的adjacency matrix
4  int lx[X], ly[Y]; // vertex labeling
5  int mx[X], my[Y]; //
6  // x各點的配對對象、y各點的配對對象
7  int q[X], *qf, *qb; // BFS queue
8  int p[X]; // BFS
9  // parent, 交錯樹之偶點, 指向上一個偶點
10 bool vx[X], vy[Y]; // 記錄是否在交錯樹上
11 int dy[Y], pdy[Y]; // 表格
12
13 void relax(int x){ // relaxation
14     for (int y=0; y<Y; ++y)
15         if (adj[x][y] != 1e9)
16             if (lx[x] + ly[y] - adj[x][y] < dy[y]){
17                 dy[y] = lx[x] + ly[y] - adj[x][y];
18                 pdy[y] = x; // 記錄好是從哪個樹葉連出去的
19             }
20 }
21
22 void reweight(){ // 調整權重、調整表格
23     int d = 1e9;
24     for (int y=0; y<Y; ++y) if (!vy[y]) d = min(d, dy[y]);
25     for (int x=0; x<X; ++x) if (vx[x]) lx[x] -= d;
26     for (int y=0; y<Y; ++y) if (vy[y]) ly[y] += d;
27     for (int y=0; y<Y; ++y) if (!vy[y]) dy[y] -= d;
28 }
29
30 void augment(int x, int y){ // 擴充路徑
31     for (int ty; x != -1; x = p[x], y = ty){
32         ty = mx[x]; my[y] = x; mx[x] = y;
33     }
34 }
35
36 bool branch1(){ // 延展交錯樹: 使用既有的等邊
37     while (qf < qb)
38         for (int x=*qf++, y=0; y<Y; ++y)
39             if (!vy[y] && lx[x] + ly[y] == adj[x][y]){
40                 vy[y] = true;
41                 if (my[y] == -1){
42                     augment(x, y);
43                     return true;
44                 }
45                 int z = my[y];
46                 *qb++ = z; p[z] = x; vx[z] = true;
47                 relax(z);
48             }
49     return false;
50 }
51
52 bool branch2(){ // 延展交錯樹: 使用新添的等邊
53     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;
55         relax(z);
56     }
57 }
58 return false;
59 }
60 int Hungarian(){
61     // 初始化vertex labeling
62     // memset(lx, 0, sizeof(lx)); // 任意值皆可
63     memset(ly, 0, sizeof(ly));
64     for (int x=0; x<X; ++x)
65         for (int y=0; y<Y; ++y)
66             lx[x] = max(lx[x], adj[x][y]);
67
68     // x側每一個點, 分別建立等邊交錯樹。
69     memset(mx, -1, sizeof(mx));
70     memset(my, -1, sizeof(my));
71     for (int x=0; x<X; ++x){
72         memset(vx, false, sizeof(vx));
73         memset(vy, false, sizeof(vy));
74         memset(dy, 0x7f, sizeof(dy));
75         qf = qb = q;
76         *qb++ = x; p[x] = -1; vx[x] = true; relax(x);
77         while (true){
78             if (branch1()) break;
79             reweight();
80             if (branch2()) break;
81         }
82     }
83     // 計算最大權完美匹配的權重
84     int weight = 0;
85     for (int x=0; x<X; ++x)
86         weight += adj[x][mx[x]];
87     return weight;
88 }

```

5.6 Dinic

```

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

```

```

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

```

5.7 Bipatirate

```

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

```

```

40             v[b].emplace_back(a);
41         }
42         isBipatirate();
43     }
44 }

```

5.8 Hungarian algorithm

```

1  const int maxn = 500+5;
2  int t, N, bn, gn, match[maxn];
3  bool visited[maxn];
4  vector<vector<int>> G(maxn);
5  struct People{
6      int h;
7      string music, sport;
8      People(){ }
9      People(int h, string music, string sport){
10         this->h = h;
11         this->music = music;
12         this->sport = sport;
13     }
14 }lef[maxn], rig[maxn];
15 bool check(People boy, People girl){
16     if(abs(boy.h - girl.h) <= 40 && boy.music ==
        girl.music && boy.sport != girl.sport) return
        true;
17     return false;
18 }
19 bool dfs(int s){
20     for(int i = 0; i < G[s].size(); ++i){
21         int v = G[s][i];
22         if(visited[v]) continue;
23         visited[v] = true;
24         if(match[v] == -1 || dfs(match[v])){
25             match[v] = s;
26             return true;
27         }
28     }
29     return false;
30 }
31 int Hungarian(){
32     int cnt = 0;
33     memset(match, -1, sizeof(match));
34     for(int i = 0; i < bn; ++i){
35         memset(visited, false, sizeof(visited));
36         if(dfs(i)) cnt++;
37     }
38     return cnt;
39 }
40 int main(){
41     cin >> t;
42     while(t--){
43         cin >> N;
44         bn = 0, gn = 0;
45         for(int i = 0; i <= N; ++i) G[i].clear();
46         int h;
47         string sex, music, sport;
48         for(int i = 0; i < N; ++i){
49             cin >> h >> sex >> music >> sport;
50             if(sex == "M") lef[bn++] = People(h,
                music, sport);
51             else rig[gn++] = People(h, music, sport);
52         }
53         for(int i = 0; i < bn; ++i){
54             for(int j = 0; j < gn; ++j)
55                 if(check(lef[i], rig[j]))
56                     G[i].emplace_back(j);
57         }
58         cout << N - Hungarian() << endl;
59     }

```

5.9 LCA

```

1  /*最低共同祖先*/
2  // 此 node 下有機類 node
3  int dfs(int node, int dep){
4      depth[node] = dep + 1;
5      if(G[node].empty()){
6          siz[node] = 1;
7          return 1;
8      }
9      int total = 1;
10     for(auto i : G[node])
11         total += dfs(i.v, dep + 1);
12     siz[node] = total;
13     return siz[node];
14 }
15 // 找出每個節點的 2^i 倍祖先
16 // 2^20 = 1e6 > 200000
17 void find_parent(){
18     for(int i = 1; i < 20; i++){
19         for (int j = 0; j < N; j++){
20             parent[j][i] =
21                 parent[parent[j][i-1]][i-1];
22 }
23 // 求兩點的LCA (利用倍增法)
24 int LCA(int a, int b){
25     if (depth[b] < depth[a]) swap(a, b);
26     if (depth[a] != depth[b]){
27         int dif = depth[b] - depth[a];
28         for (int i = 0; i < 20; i++){
29             if (dif & 1) b = parent[b][i];
30             dif >>= 1;
31         }
32     }
33     if (a == b) return a;
34     for (int i = 19; i >= 0; i--){
35         if (parent[a][i] != parent[b][i]){
36             a = parent[a][i];
37             b = parent[b][i];
38         }
39     }
40     return parent[a][0];
}

```

6 Other

6.1 Ants Colony

```

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

```

```

29             parent[j][i] =
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 }
59 int main(){
60     while(cin >> N && N){
61         memset(dist, 0, sizeof(dist));
62         memset(parent, 0, sizeof(parent));
63         memset(depth, 0, sizeof(depth));
64         memset(siz, 0, sizeof(siz));
65         for(int i = 0; i <= N; ++i){
66             G[i].clear();
67 }
68     for(int i = 1; i < N; ++i){
69         int u, w;
70         cin >> u >> w;
71         G[u].push_back({i, w});
72         parent[i][0] = u;
73 }
74     find_parent();
75     dfs(0, 0);
76     distance();
77     int s; cin >> s;
78     bool space = false;
79     for(int i = 0; i < s; ++i){
80         int a, b;
81         cin >> a >> b;
82         int lca = LCA(a, b);
83         if(space) cout << " ";
84         space = true;
85         cout << (dist[a] + dist[b]) - (dist[lca]
86             * 2);
87     }
88     cout << endl;
89 }

```

6.2 Binary codes

```

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

```

7 Math

7.1 Big Mod

```

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

```

7.2 Bubble Sort Expect Value

```

1  /* 數論 期望值算法:
2  擲一枚公平的六面骰子, 其每次「點數」的期望值是 3.5
3  E(x) = 1 * 1/6 + 2 * 1/6 + 3 * 1/6 + 4 * 1/6 + 5 *
4      1/6 + 6 * 1/6
5      = (1 + 2 + 3 + 4 + 5 + 6)/6 = 3.5
6  bubble sort 每兩兩之間交換機率是 1/2
7  總共會做 C(n, 2) 次
8  E(x) = C(n, 2) * 1/2 = (n * (n - 1))/2 * 1/2 */
9  int t, ca = 1;
10 cin >> t;
11 while(t--){
12     long long int n;
13     cin >> n;
14     cout << "Case " << ca++ << ": ";
15     // 如果 (n * (n - 1)) 可以被 4 整除
16     // 代表最後答案會是整數, 否則會是分數
17     if((n * (n - 1)) % 4){
18         cout << ( (n * (n - 1)) / 2 ) << "/2" << endl;
19     }
20     else{
21         cout << ( (n * (n - 1)) / 2 ) / 2 << endl;
22     }
23 }

```

7.3 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 }

```

7.4 How Many Os

```

1  /* 數論 */
2  int main(){
3      long long int n, m;
4      while(cin >> n >> m && (n >= 0) && (m >= 0)){
5          long long int total1 = 0, total2 = 0;
6          long long int ten = 1, tmp = n-1;
7          while(tmp >= 10){
8              if(tmp % 10 == 0){
9                  tmp /= 10;
10                 total1 += (tmp - 1) * ten + ((n-1) %
11                     ten) + 1;
12             }
13             else{
14                 tmp /= 10;
15                 total1 += tmp * ten;
16             }
17             ten *= 10;
18         }
19         ten = 1; tmp = m;
20         while(tmp >= 10){
21             if(tmp % 10 == 0){
22                 tmp /= 10;
23                 total2 += (tmp - 1) * ten + (m % ten)
24                     + 1;
25             }
26             else{
27                 tmp /= 10;
28                 total2 += tmp * ten;
29             }
30             ten *= 10;
31         }
32         if(n == 0) total1--;
33         cout << total2 - total1 << endl;
34     }
35 }

```

7.5 Number of Pairs

```

1  /* 數論
2  upper_bound ex:
3  10 20 30 30 40 50
4  upper_bound for element 30 is at index 4
5  lower_bound ex:
6  10 20 30 40 50
7  lower_bound for element 30 at index 2 */
8  int main(){
9      int t;
10     cin >> t;
11     while(t--){
12         int n, l, r;
13         vector<int> v;
14         cin >> n >> l >> r;
15         int num;
16         for(int i = 0; i < n; i++){
17             cin >> num;
18             v.emplace_back(num);
19         }
20         sort(v.begin(), v.end());
21         long long int ans = 0;
22         for(int i = 0; i < n; i++){
23             ans += (upper_bound(v.begin() + i + 1,
24                 v.end(), r - v[i]) -
25                 lower_bound(v.begin() + i + 1,
26                     v.end(), l - v[i]));

```

```

24     cout << ans << endl;
25 }
26 }

```

7.6 ORXOR

```

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

```

7.7 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     }
15 }

```

8 Function

8.1 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

```

8.2 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  }

```

8.3 map set

```

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

```

8.4 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

```

8.5 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

```


8.6 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 );

```

8.7 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

```

8.8 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()

```

8.9 sort

```

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

```

8.10 struct

```

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

```

8.11 deque

```

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

```

8.12 python template

```

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

```