

```
1: #include <stdio.h>
2: #include <vector>
3: #include <queue>
4: #define INF 2000000000000LL
5: #define pb push_back
6: using namespace std;
7:
8: /* Exemplo de uso de dijkstra, resolucao do problema "codeforces 20C - Dijkstra?"
*/
9:
10: vector <int> g[100005];
11: vector <int> w[100005];
12:
13: int pred[100005];
14: long long int d[100005];
15: int seq[100005];
16:
17: priority_queue < pair <long long int, int> > pq;
18:
19: int main(void) {
20:     int n, m;
21:     int x, y, z;
22:     int next;
23:     int viz;
24:
25:     int i;
26:     int k;
27:
28:     scanf(" %d %d", &n, &m);
29:
30:     for (i = 0; i < m; i++) {
31:         scanf(" %d %d %d", &x, &y, &z);
32:         g[x].pb(y);
33:         w[x].pb(z);
34:         g[y].pb(x);
35:         w[y].pb(z);
36:     }
37:
38:     for (i = 1; i <= n; i++) {
39:         d[i] = INF;
40:     }
41:
42:     d[1] = 0;
43:     pred[1] = 1;
44:     pq.push(make_pair(0, 1));
45:
46:     while(!pq.empty()) {
47:         long long int cost = -pq.top().first;
48:         next = pq.top().second;
49:         pq.pop();
50:
51:         if (d[next] < cost) {
52:             continue;
53:         }
54:
55:         for (i = 0; i < (int)g[next].size(); i++) {
56:             viz = g[next][i];
57:             if (d[next] + w[next][i] < d[viz]) {
58:                 pred[viz] = next;
59:                 d[viz] = d[next] + w[next][i];
60:                 pq.push(make_pair(-d[viz], viz));
61:             }
62:         }
63:     }
64:
65:     if (d[n] >= INF) {
66:         printf("-1\n");
67:     } else {
68:         k = 0;
69:         for (i = n; i != 1; i = pred[i]) {
```

```
70:         seq[k++] = i;
71:     }
72:     printf("1");
73:     for (i = k - 1; i >= 0; i--) {
74:         printf(" %d", seq[i]);
75:     }
76:     printf("\n");
77: }
78:
79: return 0;
80: }
```

```
1: #include <stdio.h>
2:
3: /* Exemplo de flood fill com DFS, solucao do problema "UVA 572 - Oil Deposits" */
4:
5: const int MAXN = 105;
6:
7: char grid[MAXN][MAXN];
8: int m, n;
9:
10: int dx[] = {-1, -1, -1, 0, 1, 1, 1, 0};
11: int dy[] = {-1, 0, 1, 1, 1, 0, -1, -1};
12:
13: int is_in (int x, int y) {
14:     return 0 <= x && x < m && 0 <= y && y < n;
15: }
16:
17: void DFS(int x, int y) {
18:     grid[x][y] = '.';
19:
20:     for (int i = 0; i < 8; i++) {
21:         int nx = x + dx[i];
22:         int ny = y + dy[i];
23:
24:         if (is_in(nx, ny) && grid[nx][ny] == '@') {
25:             DFS(nx, ny);
26:         }
27:     }
28:     return;
29: }
30:
31: int main(void) {
32:
33:     while (scanf(" %d %d", &m, &n) && m > 0) {
34:         for (int i = 0; i < m; i++) {
35:             scanf(" %s", grid[i]);
36:         }
37:
38:         int res = 0;
39:         for (int i = 0; i < m; i++) {
40:             for (int j = 0; j < n; j++) {
41:                 if (grid[i][j] == '@') {
42:                     res++;
43:                     DFS(i, j);
44:                 }
45:             }
46:         }
47:         printf("%d\n", res);
48:     }
49:     return 0;
50: }
51:
```

```
1: #include <stdio.h>
2: #include <vector>
3: #include <algorithm>
4: #define pb push_back
5: #define lli long long int
6: using namespace std;
7:
8: /* Exemplo de LCA Em  $O(N \log N, \log N)$ , solucao do problema "URI - 1135 - Colônia de
Formigas" */
9:
10: const int MAXN = (int)1e5 + 5;
11: const int MAX_LOG = 20;
12:
13: vector <int> g[MAXN];
14: vector <lli> w[MAXN];
15:
16: int par[MAXN][MAX_LOG]; // par[x][i] = ancestral de x de distancia  $2^i$ 
17: lli d[MAXN]; // d[x] = distancia da raiz ate o no x
18: int depth[MAXN]; // depth[x] = profundidade do no x
19:
20: void DFS(int node, int h, lli dist, int p = -1) {
21:     par[node][0] = p;
22:     depth[node] = h;
23:     d[node] = dist;
24:
25:     for (int i = 1; i < MAX_LOG; i++) {
26:         par[node][i] = -1;
27:         int aux = par[node][i - 1];
28:         if (aux != -1) {
29:             par[node][i] = par[aux][i - 1];
30:         }
31:     }
32:
33:     for (int i = 0; i < (int)g[node].size(); i++) {
34:         int viz = g[node][i];
35:         if (viz != p) {
36:             DFS(viz, h + 1, dist + w[node][i], node);
37:         }
38:     }
39:     return;
40: }
41:
42: int get_lca(int x, int y) {
43:     if (depth[x] < depth[y]) {
44:         swap(x, y);
45:     }
46:
47:     for (int i = MAX_LOG - 1; i >= 0; i--) {
48:         if (par[x][i] != -1 && depth[par[x][i]] >= depth[y]) {
49:             x = par[x][i];
50:         }
51:     }
52:
53:     if (x == y) {
54:         return x;
55:     }
56:
57:     for (int i = MAX_LOG - 1; i >= 0; i--) {
58:         if (par[x][i] != par[y][i]) {
59:             x = par[x][i];
60:             y = par[y][i];
61:         }
62:     }
63:     return par[x][0];
64: }
65:
66: int main(void) {
67:     int n;
68:     int a, l;
69:     int s, t;
```

```
70:     int q;
71:
72:     while (scanf("%d", &n) && n) {
73:         for (int i = 0; i < n; i++) {
74:             g[i].clear();
75:             w[i].clear();
76:         }
77:         for (int i = 1; i < n; i++) {
78:             scanf("%d %d", &a, &l);
79:             g[a].pb(i);
80:             w[a].pb(l);
81:             g[i].pb(a);
82:             w[i].pb(l);
83:         }
84:
85:         DFS(0, 0, 0);
86:         scanf("%d", &q);
87:         for (int i = 0; i < q; i++) {
88:             if (i != 0) {
89:                 printf(" ");
90:             }
91:             scanf("%d %d", &s, &t);
92:             int lca = get_lca(s, t);
93:             lli res = d[s] + d[t] - 2 * d[lca];
94:             printf("%lld", res);
95:         }
96:         printf("\n");
97:     }
98:     return 0;
99: }
```

```
1: #include <stdio.h>
2: #define MAXN 105
3: #define MOD 10000
4:
5: /* Exemplo de exponenciacao rapida de matrizes para resolucao
6:    de recorrencias lineares, solucao do problema
7:    "URI - 1713 - Teletransporte"
8: */
9:
10: int mat[MAXN][MAXN];
11: int r[MAXN][MAXN];
12: int aux[MAXN][MAXN];
13:
14: void multiply(int a[MAXN][MAXN], int b[MAXN][MAXN], int c[MAXN][MAXN], int n) {
15:     for (int i = 1; i <= n; i++) {
16:         for (int j = 1; j <= n; j++) {
17:             aux[i][j] = 0;
18:         }
19:     }
20:     for (int k = 1; k <= n; k++) {
21:         for (int i = 1; i <= n; i++) {
22:             for (int j = 1; j <= n; j++) {
23:                 aux[i][j] += b[i][k] * c[k][j];
24:                 aux[i][j] %= MOD;
25:             }
26:         }
27:     }
28:     for (int i = 1; i <= n; i++) {
29:         for (int j = 1; j <= n; j++) {
30:             a[i][j] = aux[i][j];
31:         }
32:     }
33:     return;
34: }
35:
36: void fast_exp(int p, int n) {
37:     for (int i = 1; i <= n; i++) {
38:         for (int j = 1; j <= n; j++) {
39:             r[i][j] = 0;
40:         }
41:         r[i][i] = 1;
42:     }
43:
44:     while(p > 0) {
45:         if (p & 1) {
46:             multiply(r, r, mat, n);
47:         }
48:         multiply(mat, mat, mat, n);
49:         p /= 2;
50:     }
51:     return;
52: }
53:
54: int main(void) {
55:     int n, l;
56:     int s, t;
57:     int x;
58:
59:     while(scanf(" %d %d", &n, &l) != EOF) {
60:         scanf(" %d %d", &s, &t);
61:         for (int i = 1; i <= n; i++) {
62:             for (int j = 1; j <= n; j++) {
63:                 mat[i][j] = 0;
64:             }
65:             for (int j = 0; j < 4; j++) {
66:                 scanf(" %d", &x);
67:                 mat[i][x]++;
68:             }
69:         }
70:         fast_exp(l, n);
```

```
71:         printf("%d\n", r[s][t]);
72:     }
73:     return 0;
74: }
```

```
1: #include <stdio.h>
2: #include <queue>
3: #define MAXN 50005
4: #define INF 0x3f3f3f3f
5: using namespace std;
6:
7: /* Solucao do problema SUMS da POI http://main.edu.pl/en/archive/oi/10/sum */
8:
9: int v[MAXN];
10: int d[MAXN];
11:
12: int main(void) {
13:     int n;
14:     int k;
15:     int b;
16:     priority_queue <pair <int, int> > pq;
17:
18:     scanf(" %d", &n);
19:     for (int i = 0; i < n; i++) {
20:         scanf(" %d", &v[i]);
21:     }
22:     d[0] = 0;
23:     for (int i = 1; i < v[0]; i++) {
24:         d[i] = INF;
25:     }
26:     pq.push(make_pair(0, 0));
27:     while(!pq.empty()) {
28:         int cost = pq.top().first;
29:         int next = pq.top().second;
30:         pq.pop();
31:
32:         if (d[next] < cost) {
33:             continue;
34:         }
35:         for (int i = 1; i < n; i++) {
36:             int viz = (next + v[i]) % v[0];
37:             if (d[viz] > d[next] + v[i]) {
38:                 d[viz] = d[next] + v[i];
39:                 pq.push(make_pair(-d[viz], viz));
40:             }
41:         }
42:     }
43:     scanf(" %d", &k);
44:     while(k--) {
45:         scanf(" %d", &b);
46:         if (d[b % v[0]] <= b) {
47:             printf("TAK\n");
48:         } else {
49:             printf("NIE\n");
50:         }
51:     }
52:     return 0;
53: }
```