```cpp
/*****centroid decomposition*******/
using namespace std;
using namespace __gnu_pbds;
inline int add(int _a, int _b, int md){
//  _a = (_a + md) % md;
//  _b = (_b + md) % md;
   if(_a < 0){ _a += md; }
   if(_b < 0){ _b += md; }
   if(_a + _b >= md){ return _a + _b - md; }
   return _a + _b;
}
inline int mul(int _a, int _b, int md){
   if(_a < 0){ _a += md; }
   if(_b < 0){ _b += md; }
   return ((ll)((ll)_a * (ll)_b)) % md;
}
struct chash {
    int operator()(pair <int, int> x) const { return x.first* 31 + x.second; }
};
int q;
const int base[] = {229, 233}, mod[] = {417565807, 640663963};
int f_h[2][2 * 1000010], r_h[2][2 * 1000010], len, P[2][2 * 1000010];
char str[2 * 1000010];
vector < pair <int, char > > adj[SZ];
bool mark_cen[SZ];
int parent[SZ], sbtr[SZ], n, centroid, par_cen[SZ];
vector <int> adj_cen[SZ];
void dfs0(int src, int par){
   int i, j, u; sbtr[src] = 1, parent[src] = par;
   for0(i, adj[src].size()){
      u = adj[src][i].first;
      if(u != par){ dfs0(u, src), sbtr[src] += sbtr[u]; }
   }
}
int get_centroid(int src, int par){
   int i, j, u, mx = -1, bg;
   bool is_cen = true;
   for0(i, adj[src].size()){
      u = adj[src][i].first;
      if(u != par){
         if(sbtr[u] > n / 2){ is_cen = false; }
         if(sbtr[u] > mx){ mx = sbtr[u], bg = u; }
      }
   }
   if(is_cen && n - sbtr[src] <= n / 2){ return src; }
   return get_centroid(bg, src);
}
void get_centroid(int cnn, int root, int src, int par){
   int i, j, u, mx = -1, bg;
   bool is_cen = true;
   if(src == centroid){
      for0(i, adj[src].size()){
         u = adj[src][i].first;
         if(!mark_cen[u] && u != par){ get_centroid(src, u, u, src); }
      }
      return;
   }
   for0(i, adj[src].size()){
      u = adj[src][i].first;
      if(u != par && !mark_cen[u]){
         if(sbtr[u] > sbtr[root] / 2){ is_cen = false; }
         if(sbtr[u] > mx){ mx = sbtr[u], bg = u; }
      }
   }
   if(mx == -1 || (is_cen && sbtr[root] - sbtr[src] <= sbtr[root] / 2)){
      adj_cen[cnn].push_back(src), adj_cen[src].push_back(cnn);
      mark_cen[src] = true;
      int p = parent[src];
      while(!mark_cen[p]){ sbtr[p] -= sbtr[src], p = parent[p]; }
      for0(i, adj[src].size()){
         u = adj[src][i].first;
         if(u != par && !mark_cen[u]){ get_centroid(src, u, u, src); }
```

```
        }
        if(mx != -1 && !mark_cen[root]){ get_centroid(src, root, root, parent[root]); }
    }
    else{ get_centroid(cnn, root, bg, src); }
}
void decompose(){
    int i, j;
    dfs0(0, -1), centroid = get_centroid(0, -1), dfs0(centroid, -1);
    for0(i, n + 2){ mark_cen[i] = false, adj_cen[i].clear(); }
    mark_cen[centroid] = true;
    get_centroid(centroid, centroid, centroid, -1);
    for0(i, n + 2){ mark_cen[i] = false; }
}
gp_hash_table < pair <int, int>, int, chash> mp[SZ], mp_child[SZ];
gp_hash_table < int, pair <int, int > > hash_mp[SZ];
gp_hash_table <int, int> dis_mp[SZ];
void dfs2(int cen, int src, int root, int par, int d, int h0, int h1, bool tick){
    int i, j, sz = adj_cen[src].size(), cnt = 0, nh0, nh1;
    int w, u;
    mp[cen][mpr(h0, h1)]++; dis_mp[cen][src] = d; hash_mp[cen][src] = mpr(h0, h1);
    if(tick){ mp_child[root][mpr(h0, h1)]++; }
    cnt = root == centroid ? 0 : 1;
    for0(i, adj[src].size()){
        u = adj[src][i].first;
        w = adj[src][i].second - 'a' + 1;
        nh0 = add(w, mul(h0, base[0], mod[0]), mod[0]), nh1 = add(w, mul(h1, base[1],
mod[1]), mod[1]);
        if(!tick){
            if(u != par && !mark_cen[u]){
                dfs2(cen, u, adj_cen[src][cnt], src, d + 1, nh0, nh1, true); cnt++;
            }
            else if(u == par && !mark_cen[u]){
                dfs2(cen, u, adj_cen[src][sz - 1], src, d + 1, nh0, nh1, true);
            }
        }
        else if(u != par && !mark_cen[u]){
            dfs2(cen, u, root, src, d + 1, nh0, nh1, true);
        }
    }
}
void dfs1(int src, int par){
    int i, j, u; mark_cen[src] = true; par_cen[src] = par;
    mp[src].clear(); dis_mp[src].clear(); hash_mp[src].clear();
    for0(i, adj_cen[src].size()){
        u = adj_cen[src][i];
        if(u != par && !mark_cen[u]){ mp_child[u].clear();}
    }
    dfs2(src, src, src, parent[src], 0, 0, 0, false);
    for0(i, adj_cen[src].size()){
        u = adj_cen[src][i];
        if(u != par && !mark_cen[u]){ dfs1(u, src); }
    }
}
void input(){
    int i, j, u, v;
    char w;
    sii(n, q);
    for0(i, n + 2){ adj[i].clear(); }
    for0(i, n - 1){
        scanf("%d %d %c", &u, &v, &w), u--, v--;
        adj[u].push_back(mpr(v, w)), adj[v].push_back(mpr(u, w));
    }
    decompose();
}
void make_hash(){
    int i, j;
    for0(i, 2){
        for(j = len - 1; j >= 0; j--){
            f_h[i][j] = j == len - 1 ? str[j] - 'a' + 1 :
                add(str[j] - 'a' + 1, mul(f_h[i][j + 1], base[i], mod[i]), mod[i]);
        }
        for(j = 0; j < len; j++){
```

```
            r_h[i][j] = j == 0 ? str[j] - 'a' + 1 :
               add(str[j] - 'a' + 1, mul(r_h[i][j - 1], base[i], mod[i]), mod[i]);
         }
      }
   //   para(i, 0, len - 1, f_h[0]);
   //   para(i, 0, len - 1, r_h[0]);
   }
   pair <int, int> for_hash(int l, int r){
      if(l > r){ return mpr(0, 0); }
      if(r == len - 1){ return mpr(f_h[0][l], f_h[1][l]); }
      return mpr(add(f_h[0][l], -mul(P[0][r - l + 1], f_h[0][r + 1], mod[0]), mod[0])
                 , add(f_h[1][l], -mul(P[1][r - l + 1], f_h[1][r + 1], mod[1]), mod[1]));
   }
   pair <int, int> rev_hash(int l, int r){
      if(l > r){ return mpr(0, 0); }
      if(l == 0){ return mpr(r_h[0][r], r_h[1][r]); }
      return mpr(add(r_h[0][r], -mul(P[0][r - l + 1], r_h[0][l - 1], mod[0]), mod[0])
                 , add(r_h[1][r], -mul(P[1][r - l + 1], r_h[1][l - 1], mod[1]), mod[1]));
   }
   int query(int u){
      int i, j, p = u, d, prv;
      pair <int, int> f, r;
      int ret = 0;
      for(p = u, prv = u; p != -1; ){
         if(p == u){
            r = rev_hash(0, len - 1);
            ret += mp[p][r];
         }
         else{
            d = dis_mp[p][u];
            if(d <= len){
               f = for_hash(0, d - 1);
               if(f == hash_mp[p][u]){
                  r = rev_hash(d, len - 1);
                  ret += mp[p][r] - mp_child[prv][r];
               }
            }
         }
         prv = p; p = par_cen[p];
      }
      return ret;
   }
   void solve(){
      int i, j, u, ret;
      dfs1(centroid, -1);
      for0(i, q){
         scanf("%d %s", &u, &str); u--; len = strlen(str);
         make_hash();
         ret = query(u);
         pi(ret); nl;
      }
      for0(i, n + 2){ adj[i].clear(), adj_cen[i].clear(); }
   }
   void pre_process(){
      int i, j;
      for0(i, 2){
         for(j = 1, P[i][0] = 1; j <= 2000005; j++){
            P[i][j] = mul(P[i][j - 1], base[i], mod[i]);
         }
      }
   //   para(i, 0, 10, P[0]);
   }
   int main(){
   //  freopen("input.txt","r",stdin);
   //    freopen("output.txt", "w", stdout);
      pre_process();
     int cs, ts;
     si(ts);
     for0(cs, ts){
         input();
         printf("Case %d:\n", cs + 1);
         solve();
```

```
  }
}


/********suffix automata*********/
int add(int _a, int _b, int md){
   if(_a < 0){ _a += md; }
   if(_b < 0){ _b += md; }
   if(_a + _b >= md){ return _a + _b - md; }
   return _a + _b;
}
int mul(int _a, int _b, int md){
   if(_a < 0){ _a += md; }
   if(_b < 0){ _b += md; }
   return ((ll)((ll)_a * (ll)_b)) % md;
}
vector <int> adj[SZ];
struct state{
    int len, link;
    gp_hash_table <char, int> next;
};
state st[SZ * 2];
int to_state = 0, last, occ[SZ * 2];
//occ should be initialized for every testcase
void sa_init(){
   to_state = 0;
   st[0].len = 0;
   st[0].link = -1;
   to_state++;
   last = 0;
}
void sa_extend(char c) {
    int cur = to_state++;
    st[cur].len = st[last].len + 1; occ[cur] = 1;
    st[cur].next.clear(); adj[cur].clear();
    int p = last;
    while (p != -1 && st[p].next.find(c) == st[p].next.end()) {
        st[p].next[c] = cur;
        p = st[p].link;
    }
    if (p == -1) { st[cur].link = 0; }
    else{
        int q = st[p].next[c];
        if(st[p].len + 1 == st[q].len){ st[cur].link = q; }
        else{
            int clone = to_state++;
            st[clone].len = st[p].len + 1;
            st[clone].next = st[q].next;
            st[clone].link = st[q].link;
            occ[clone] = 0; adj[clone].clear();
            while (p != -1 && st[p].next[c] == q) {
                st[p].next[c] = clone;
                p = st[p].link;
            }
            st[q].link = st[cur].link = clone;
        }
    }
    last = cur;
}
char str[SZ];
int q, A, B, n, q_len;
char q_str[SZ];
const int magic = 300;
void input(){
   int i, j;
   ss(str); n = strlen(str);
   siii(q, A, B);
}
void dfs(int src){
   int i, j, u;
   for0(i, adj[src].size()){
      u = adj[src][i];
```

```
        dfs(u);
        occ[src] += occ[u];
    }
}
void rebuild(int offst){
    int i, j, u, v;
    sa_init();
    for0(i, n + offst){ sa_extend(str[i]); }
    for0(i, to_state){
        u = i, v = st[i].link;
        if(v != -1){ adj[v].push_back(u); }
    }
    dfs(0);
}
int prefix[SZ];
int do_brute(int from, int to){
    int i, j, cnt = 0, k;
    if(to - from + 1 < q_len){ return 0; }
    for(k = -1, i = 1, prefix[0] = 0; i < q_len; i++){
        while(k >= 0 && q_str[k + 1] != q_str[i]){ k = prefix[k] - 1; }
        if(q_str[k + 1] == q_str[i]){ k++; }
        prefix[i] = k + 1;
    }
    for(k = -1, i = from; i <= to; i++){
        while(k >= 0 && q_str[k + 1] != str[i]){ k = prefix[k] - 1; }
        if(q_str[k + 1] == str[i]){ k++; }
        if(k + 1 == q_len){ cnt++; k = prefix[k] - 1; }
    }
    return cnt;
}
int occurance(){
    int i, j, t = 0;
    for(i = 0; i < q_len; i++){
        if(st[t].next.find(q_str[i]) != st[t].next.end()){ t = st[t].next[q_str[i]]; }
        else{ return 0; }
    }
    return occ[t];
}
void solve(){
    int i, j, v, ret = 0;
    rebuild(0);
    for(i = 0; i < q; i++){
        ss(q_str); q_len = strlen(q_str);
        ret = 0; ret += do_brute(max(n + (i / magic) * magic - q_len + 1, 0), n + i - 1);
        ret += occurance();
        pi(ret); nl;
        v = add(B, mul(A, ret, 26), 26) + 'a';
        str[n + i] = (char)v;
        if((i + 1) % magic == 0){ rebuild(i + 1); }
    }
}
int main(){
//    freopen("input.txt","r",stdin);
//    freopen("output.txt", "w", stdout);
    input();
    solve();
}


/*******all number phi(n) = x*****backtracking solution*****/
using namespace std;
bool status[SZ];
ll prime[SZ], p_cnt = 0;
void siv(){
    ll N = 1000005, i, j;
    ll sq = sqrt(N);
    for(i = 4; i <= N; i += 2){ status[i] = true; }
    for(i = 3; i <= sq; i+= 2){
        if(status[i] == false){
            for(j = i * i; j <= N; j += i){ status[j] = true; }
        }
    }
```

```
360       status[1] = true;
361       for1(i, N){ if(!status[i]){ prime[p_cnt++] = i; } }
362    }
363    ll add(ll _a, ll _b){
364       _a = (_a + MOD) % MOD;
365       _b = (_b + MOD) % MOD;
366       return (_a + _b) % MOD;
367    }
368    ll mul(ll _a, ll _b){
369       _a = (_a + MOD) % MOD;
370       _b = (_b + MOD) % MOD;
371       return ((ll)((ll)_a * (ll)_b)) % MOD;
372    }
373    ll phi;
374    vector <ll> res;
375    vector < pair <ll, ll > > factorize(ll x){
376       ll i, j;
377       vector < pair <ll, ll > > ret;
378       for(i = 0; i < p_cnt && prime[i] <= x / prime[i]; i++){
379          if(x % prime[i] == 0){
380             ret.push_back(mpr(prime[i], 0));
381             while(x % prime[i] == 0){ ret.back().second++; x /= prime[i]; }
382          }
383       }
384       if(x != 1){ ret.push_back(mpr(x, 1)); }
385       return ret;
386    }
387    bool is_prime(ll x){
388       ll i, j;
389       if(x <= 1000000){ return (!status[x]); }
390       for(i = 0; i < p_cnt && prime[i] <= x / prime[i]; i++){
391          if(x % prime[i] == 0){ return false; }
392       }
393       return true;
394    }
395    void div_search(ll pos, ll d, ll thresh, vector <pair <ll, ll > > & factor, vector <ll>
       &divisor){
396       if(thresh == 0){ return; }
397       ll i, j, ret = d;
398       if(pos == factor.size()){ divisor.push_back(d); return; }
399       div_search(pos + 1, d, thresh, factor, divisor);
400       for(i = 0; i < factor[pos].second; i++){
401          if(d <= thresh / factor[pos].first){
402             d *= factor[pos].first;
403             div_search(pos + 1, d, thresh, factor, divisor);
404          }
405       }
406    }
407    void F(ll x, ll y, ll mx_prime){
408       if(x == 1){
409          res.push_back(y);
410          if(mx_prime > 2){ res.push_back(2ll * y); }
411          return;
412       }
413       ll i, j, d, u, v;
414       vector < pair <ll, ll > > factor = factorize(x);
415       vector <ll> divisor;
416       div_search(0, 1, mx_prime - 2, factor, divisor);
417       sort(divisor.begin(), divisor.end());
418    //    pi(x), ps("-->"); sp; pvec(i, divisor);
419       for0(i, divisor.size()){
420          d = divisor[i];
421          if(is_prime(d + 1)){
422             u = x / d, v = y;
423             while(u % (d + 1) == 0){ u /= (d + 1), v *= (ll)(d + 1); }
424             v *= (ll)(d + 1);
425             F(u, v, d + 1);
426          }
427       }
428    }
429    void solve(){
430       ll i, j;
```

```
431      res.clear();
432      F(phi, 1ll, INF);
433      if(res.size() == 0){ ps("No solution.\n"); return; }
434      sort(res.begin(), res.end());
435      for0(i, res.size()){
436         if(i != 0){ sp; }
437         pl(res[i]);
438      } nl;
439   }
440   int main(){
441   //    freopen("input.txt","r",stdin);
442   //    freopen("output.txt", "w", stdout);
443      ll i, j;
444      siv();
445      while(si(phi) == 1){
446         solve();
447      }
448   }
449
450
451   /****number of connected components in the complement graph*****/
452   int n, m, nd_cnt = 0, k, frbd_cnt = 0;
453   vector <int> rev[SZ], adj[3 * SZ];
454   pair <int, int> tree[2 * SZ];
455   bool bad[SZ], vis[2 * SZ], color[3 * SZ], global;
456   int compo_cnt = 0;
457   queue <int> q;
458   void input(){
459      int i, j, u, v;
460      siii(n, m, k);
461      frbd_cnt = 0;
462      for0(i, m){
463         sii(u, v); u--, v--;
464         if(u == 0 || v == 0){ if(u){ bad[u] = true; } else{ bad[v] = true; } frbd_cnt++;
   continue; }
465         rev[u].push_back(v), rev[v].push_back(u);
466      }
467   }
468   void make_tree(int lo, int hi, int iter){
469      int mid = (lo + hi) >> 1;
470      if(lo == hi){ return; }
471      else{
472         nd_cnt++, tree[iter].first = nd_cnt - 1;
473         make_tree(lo, mid, tree[iter].first);
474         nd_cnt++, tree[iter].second = nd_cnt - 1;
475         make_tree(mid + 1, hi, tree[iter].second);
476      }
477   }
478   inline void add_edge(int u, int v){
479      adj[u].push_back(v);
480      adj[v].push_back(u);
481   }
482   void add_link(int lo, int hi, int iter, int l, int r, int v){
483      if(l > r){ return; }
484      int mid = (lo + hi) >> 1, i, j;
485      if(l <= lo && r >= hi){
486         if(!vis[iter]){
487            for(vis[iter] = true, i = lo; i <= hi; i++){
488               add_edge(i, iter + n);
489            }
490         }
491         if(!(v >= lo && v <= hi)){ add_edge(v, iter + n); }
492         return;
493      }
494      else if(l > hi || r < lo){ return; }
495      else{
496         add_link(lo, mid, tree[iter].first, l, r, v);
497         add_link(mid + 1, hi, tree[iter].second, l, r, v);
498      }
499   }
500   void bfs(int src){
501      int i, j, u, v;
```

```
    q.push(src);
    while(!q.empty()){
        u = q.front(), q.pop();
        if(!bad[u]){ global = true; }
        for0(i, adj[u].size()){
            v = adj[u][i];
            if(!color[v]){ color[v] = true; q.push(v); }
        }
    }
}
void solve(){
    if(n - 1 - frbd_cnt < k){ ps("impossible\n"); return; }
    int i, j, pv;
    nd_cnt = 1, make_tree(0, n - 1, 0);
    for1(i, n - 1){ rev[i].push_back(0), sort(rev[i].begin(), rev[i].end()); }
    for1(i, n - 1){
        for(pv = -1, j = 0; j < rev[i].size(); j++){
            add_link(0, n - 1, 0, pv + 1, rev[i][j] - 1, i), pv = rev[i][j];
        }
        add_link(0, n - 1, 0, pv + 1, n - 1, i);
    }
    int x = 0;
    compo_cnt = 0;
    for1(i, n - 1){
        if(!color[i]){
            global = false;
            bfs(i);
            if(!global){ ps("impossible\n"); return; }
            compo_cnt++;
        }
    }
    if(compo_cnt > k){ ps("impossible\n"); return; }
    ps("possible\n");
}
int main(){
//    freopen("input.txt","r",stdin);
//    freopen("output.txt", "w", stdout);
    input();
    solve();
}


/***************ntt**********/
const int N = 1 << 20, mod = 786433, g = 10;
int rev[N], w[N], inv_n;
int bigMod(int v, int p){
    if(p == 0){ return 1; }
    int ret = bigMod(v, p / 2);
    if(p % 2 == 0){ return mul(ret, ret); }
    else{ return mul(ret, mul(ret, v)); }
}
void prepare(int &n) {
    int sz = 31 - __builtin_clz(n); sz = abs(sz);
    int r = bigMod(g, (mod - 1) / n);
    inv_n = bigMod(n, mod - 2);
    w[0] = w[n] = 1;
    for(int i = 1; i < n; ++i) w[i] = (ll)w[i - 1] * r % mod;
    for(int i = 1; i < n; ++i) rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (sz - 1));
}
void ntt(int *a, int n, int dir) {
    for(int i = 1; i < n - 1; ++i)
        if(i < rev[i]) swap(a[i], a[rev[i]]);
    for(int m = 2; m <= n; m <<= 1) {
        for(int i = 0; i < n; i += m) {
            for(int j = 0; j < (m >> 1); ++j) {
                int &u = a[i + j], &v = a[i + j + (m >> 1)];
                int t = (ll)v * w[dir ? n - n / m * j : n / m * j] % mod;
                v = u - t < 0 ? u - t + mod : u - t;
                u = u + t >= mod ? u + t - mod : u + t;
            }
        }
    } if(dir) for(int i = 0; i < n; ++i) a[i] = (ll)a[i] * inv_n % mod;
```

```
574   }
575   int f_a[N], f_b[N];
576   vector <int> multiply(vector <int> a, vector <int> b){
577       int sz = 1, na = (int)(a.size()), nb = (int)(b.size());
578       while(sz < na + nb - 1){ sz <<= 1; }
579       prepare(sz);
580       for (int i = 0; i < sz; ++i) f_a[i] = i < na ? a[i] : 0;
581       for (int i = 0; i < sz; ++i) f_b[i] = i < nb ? b[i] : 0;
582       ntt(f_a, sz, 0); ntt(f_b, sz, 0);
583       for (int i = 0; i < sz; ++i) f_a[i] = (ll) f_a[i] * f_b[i] % MOD;
584       ntt(f_a, sz, 1); return vector <int> (f_a, f_a + sz);
585   }
586   // primitive root, finding the number with order p-1
587   int primitive_root(int p) {
588       vector<int> factor;
589       int tmp = p - 1;
590       for(int i = 2; i * i <= tmp; ++i) {
591           if (tmp % i == 0) {
592               factor.push_back(i);
593               while (tmp % i == 0) tmp /= i;
594           }
595       }
596       if(tmp != 1) factor.push_back(tmp);
597       for(int root = 1; ; ++root) {
598           bool flag = true;
599           for(int i = 0; i < factor.size(); ++i) {
600               if(Pow(root, (p - 1) / factor[i], p) == 1) {
601                   flag = false;
602                   break;
603               }
604           }
605           if (flag) return root;
606       }
607   }
608
609
610   /***************bld*********************/
611   inline int add(int _a, int _b){
612       if(_a < 0){ _a = _a + MOD; }
613       if(_b < 0){ _b = _b + MOD; }
614       if(_a + _b > MOD){ return _a + _b - MOD; }
615       return _a + _b;
616   //    return (_a + _b) % MOD;
617   }
618   inline int mul(int _a, int _b){
619   //    _a = (_a + MOD) % MOD;
620   //    _b = (_b + MOD) % MOD;
621       if(_a < 0){ _a = _a + MOD; }
622       if(_b < 0){ _b = _b + MOD; }
623       return (_a * 1ll * _b) % MOD;
624   }
625   struct matrix{
626       int a, b, c, d;
627       matrix(){}
628       matrix(int _a, int _b, int _c, int _d){ a = _a, b = _b, c = _c, d = _d; }
629       matrix operator +(const matrix & o) const{
630           return matrix(add(a, o.a), add(b, o.b), add(c, o.c), add(d, o.d));
631       }
632       matrix operator -(const matrix & o) const{
633           return matrix(add(a, -o.a), add(b, -o.b), add(c, -o.c), add(d, -o.d));
634       }
635       matrix operator *(const matrix & o) const{
636           return matrix(add(mul(a, o.a), mul(b, o.c)), add(mul(a, o.b), mul(b, o.d)),
      add(mul(c, o.a), mul(d, o.c)), add(mul(c, o.b), mul(d, o.d)));
637       }
638       bool operator !=(const matrix & o) const{
639           if(a != o.a || b != o.b || c != o.c || d != o.d){ return true; }
640           return false;
641       }
642       void print(){
643           pii(a, b); nl;
644           pii(c, d); nl;
```

```
645         }
646     }fib, fibP[SZ], unit, zero, invFib;
647     struct seg{
648         matrix t, sum, lz;
649         seg(){}
650         seg(matrix _t, matrix _sum, matrix _lz){ t = _t, sum = _sum, lz = _lz; }
651     }tree[4 * SZ + 10];
652     int n, q;
653     vector <int> adj[SZ];
654     int hNo[SZ], hHd[SZ], hSz[SZ], hPos[SZ], cLink[SZ], ht[SZ], mxht;
655     int sbtr[SZ], hCnt = 0, ndCnt = 0, mark[SZ];
656     vector <int> vrtx;
657     matrix matExpo(matrix z, ll p){
658         ll i, j; matrix ret = unit;
659         for(i = p; i != 0; i >>= 1){ if(i % 2 == 1){ ret = ret * z; } z = z * z; }
660         return ret;
661     }
662     void powering(){
663         int i, j;
664         unit = matrix(1, 0, 0, 1), zero = matrix(0, 0, 0, 0);
665         fib = matrix(1, 1, 1, 0); invFib = matrix(0, 1, 1, MOD - 1);
666         for(i = 2, fibP[0] = unit, fibP[1] = fib; i <= mxht; i++){
667             fibP[i] = fibP[i - 1] * fib;
668         }
669     }
670     void dfs(int src, int par){
671         int i, j; sbtr[src] = 1;
672         for0(i, adj[src].size()){
673             int u = adj[src][i];
674             if(u != par){ dfs(u, src); sbtr[src] += sbtr[u]; }
675         }
676     }
677     void bldHld(int src, int par, bool f, int d){
678         int i, j, bg = -1, mx = -1, u; vrtx.push_back(src); ht[src] = d, mxht = max(mxht,
    ht[src]);
679         if(f){ hCnt++, hHd[hCnt - 1] = src, hSz[hCnt - 1] = 0, cLink[hCnt - 1] = par; }
680         hNo[src] = hCnt - 1, hPos[src] = ndCnt++, hSz[hCnt - 1]++;
681         for0(i, adj[src].size()){
682             u = adj[src][i];
683             if(u != par){ if(sbtr[u] > mx){ mx = sbtr[u], bg = u; } }
684         }
685         if(bg != -1){ bldHld(bg, src, false, d + 1); }
686         for0(i, adj[src].size()){
687             u = adj[src][i];
688             if(u != par && u != bg){ bldHld(u, src, true, d + 1); }
689         }
690     }
691     void makeHld(){
692         int i, j;mxht = -1, dfs(0, -1);
693         hCnt = 0, ndCnt = 0, vrtx.clear();
694         bldHld(0, -1, true, 0);
695         for0(i, hCnt){ mark[i] = -1; }
696     //    nvec(i, vrtx);
697     //    for0(i, vrtx.size()){
698     //        int idx = vrtx[i];
699     //        niii(idx, hNo[idx], hPos[idx]); sp; pi(cLink[hNo[idx]]); nl;
700     //    }
701     }
702     int getLca(int u, int v){
703         int i, j, nd = u;
704         vector <int> vec;
705         while(1){
706             int no = hNo[nd]; mark[no] = nd; vec.push_back(no);
707             if(no == 0){ break; } nd = cLink[no];
708         }
709         nd = v;
710         while(1){
711             int no = hNo[nd];
712             if(mark[no] != -1){
713                 int ret = mark[no]; for0(i, vec.size()){ mark[vec[i]] = -1; }
714                 if(ht[nd] < ht[ret]){ return nd; }
715                 else{ return ret; }
```

```
716             }
717             nd = cLink[no];
718         }
719         return -1;
720     }
721     void input(){
722         int i, j;
723         sii(n, q);
724         for(i = 1; i <= n - 1; i++){
725             int u, v; si(u); v = i + 1, u--, v--;
726             adj[u].push_back(v), adj[v].push_back(u);
727         }
728     }
729     void mrg(int iter, bool f){
730         if(!f){ tree[iter].t = tree[2 * iter + 1].t + tree[2 * iter + 2].t; }
731         else{ tree[iter].sum = tree[2 * iter + 1].sum + tree[2 * iter + 2].sum; }
732     }
733     void makeTree(int lo, int hi, int iter){
734         int mid = (lo + hi) >> 1;
735         if(lo == hi){
736             int h = ht[vrtx[lo]];
737             tree[iter] = seg(fibP[h], zero, zero);
738             return;
739         }
740         else if(lo < hi){
741             makeTree(lo, mid, 2 * iter + 1);
742             makeTree(mid + 1, hi, 2 * iter + 2);
743             mrg(iter, false);
744         }
745     }
746     void lazyUp(int iter, matrix & z){
747         tree[iter].lz = tree[iter].lz + z;
748         tree[iter].sum = tree[iter].sum + tree[iter].t * z;
749     }
750     void pushDown(int iter){
751         if(tree[iter].lz != zero){
752             lazyUp(2 * iter + 1, tree[iter].lz);
753             lazyUp(2 * iter + 2, tree[iter].lz);
754             tree[iter].lz = zero;
755         }
756     }
757     void update(int lo, int hi, int iter, int l, int r, matrix & z){
758         int mid = (lo + hi) >> 1;
759     //    pii(lo, hi); nl;
760         if(l > r){ return; }
761         if(l <= lo && r >= hi){ lazyUp(iter, z); return; }
762         else if(l > hi || r < lo){ return; }
763         else{
764             pushDown(iter);
765             update(lo, mid, 2 * iter + 1, l, r, z);
766             update(mid + 1, hi, 2 * iter + 2, l, r, z);
767             mrg(iter, true);
768         }
769     }
770     matrix query(int lo, int hi, int iter, int l, int r){
771     //    pii(lo, hi); nl;
772         int mid = (lo + hi) >> 1;
773         if(l > r){ return zero; }
774         if(l <= lo && r >= hi){ return tree[iter].sum; }
775         else if(l > hi || r < lo){ return zero; }
776         else{
777             pushDown(iter);
778             matrix retl = query(lo, mid, 2 * iter + 1, l, r);
779             matrix retr = query(mid + 1, hi, 2 * iter + 2, l, r);
780             mrg(iter, true);
781             return retl + retr;
782         }
783     }
784     matrix pathQuery(int u, int v){
785         int i, j, hd;
786         matrix ret = zero;
787         while(1){
```

```
788             if(ht[hHd[hNo[u]]] < ht[hHd[hNo[v]]]){ swap(u, v); }
789     //        pii(u, v); nl;
790             if(hNo[u] == hNo[v]){
791                 if(hPos[u] < hPos[v]){ swap(u, v); }
792                 ret = ret + query(0, n - 1, 0, hPos[v], hPos[u]);
793                 return ret;
794             }
795             hd = hHd[hNo[u]];
796             ret = ret + query(0, n - 1, 0, hPos[hd], hPos[u]);
797             u = cLink[hNo[u]];
798         }
799         return zero;
800     }
801     void solve(){
802         makeHld();
803         powering();
804         makeTree(0, n - 1, 0);
805         int i, j, u, v;
806         matrix ret;
807         ll k;
808         char tp;
809         for0(i, q){
810             scanf(" %c", &tp);
811             if(tp == 'U'){
812                 si(u), sl(k), u--;
813                 if(k - (ll)ht[u] - 1ll < 0){ ret = matExpo(invFib, abs(k - (ll)ht[u] -
    1ll)); }
814                 else {ret = matExpo(fib, k - (ll)ht[u] - 1); }
815                 update(0, n - 1, 0, hPos[u], hPos[u] + sbtr[u] - 1, ret);
816             }
817             else{
818                 sii(u, v), u--, v--;
819                 ret = pathQuery(u, v);
820                 pi(ret.a); nl;
821             }
822         }
823     }
824     int main(){
825     //    freopen("input.txt","r",stdin);
826     //    freopen("output.txt", "w", stdout);
827         int i, j;
828         input();
829         solve();
830     }
831
832
833     /******floyd warshal maximizing multiplication**********/
834     int n, m;
835     string vrtx[SZ];
836     map < string, int> mp;
837     double adj[SZ][SZ];
838     void input(){
839        int i, j, u, v;
840        double w;
841        string str_u, str_v;
842        for0(i, n){
843           cin >> vrtx[i];
844        }
845        for0(i, n){ mp[vrtx[i]] = i; }
846        si(m);
847        mem(adj, 0); for0(i, n){ adj[i][i] = 1.0; }
848        for0(i, m){
849           cin >> str_u >> w >> str_v;
850           u = mp[str_u], v = mp[str_v];
851           adj[u][v] = w;
852        }
853     }
854     bool floyd_warshal(){
855        int i, j, k;
856        for0(k, n){
857           for0(i, n){
858              for0(j, n){
```

```
            if(adj[i][k] * adj[k][j] > adj[i][j]){
               adj[i][j] = adj[i][k] * adj[k][j];
            }
         }
      }
   }
   for0(i, n){
      if(adj[i][i] > 1.0){ return true; }
   }
   return false;
}
void solve(){
   int i, j;
   bool f = floyd_warshal();
//   ppara(i, j, n, n, adj);
   if(f){ ps("Yes\n"); }
   else{ ps("No\n"); }
}
int main(){
//    freopen("input.txt","r",stdin);
//    freopen("output.txt", "w", stdout);
   int cs = 0;
   while(si(n) == 1){
      if(n == 0){ break; }
      input();
      printf("Case %d: ", cs + 1);
      solve();
      cs++;
   }
}


/******extended euclid finding all solution in range ******/
ll gcd(ll a, ll b, ll &x, ll &y){
   if(a == 0){ x = 0, y = 1; return b; }
   ll x1, y1;
   ll d = gcd(b % a, a, x1, y1);
   x = y1 - (b / a) * x1;
   y = x1;
   return d;
}
bool find_any_solution(ll a, ll b, ll c, ll &x0, ll &y0, ll &g){
   g = gcd(abs(a), abs(b), x0, y0);
   if(c % g != 0){ return false; }
   x0 *= c / g;
   y0 *= c / g;
   if(a < 0){ x0 *= -1; }
   if(b < 0){ y0 *= -1; }
   return true;
}
ll dv(ll a, ll b, bool f){
   if(((a < 0) ^ (b < 0))){
      if(!f){ return -((abs(a) + abs(b) - 1) / abs(b)); }
      else{ return -(abs(a) / abs(b)); }
   }
   else{
      if(!f){ return abs(a) / abs(b); }
      else{ return (abs(a) + abs(b) - 1) / abs(b); }
   }
}
ll find_all_solution(ll a, ll b, ll c, ll min_x, ll max_x, ll min_y, ll max_y){
   ll x, y, g;
   if(!find_any_solution(a, b, c, x, y, g)){ return -INF; }
   ll kx_min, kx_max, ky_min, ky_max;
   if(b / g >= 0){
      kx_min = dv(min_x - x, b / g, true);
      kx_max = dv(max_x - x, b / g, false);
   }
   else{
      kx_min = dv(max_x - x, b / g, true);
      kx_max = dv(min_x - x, b / g, false);
   }
```

```
931      if(kx_max < kx_min){ return -INF; }
932
933      if(a / g >= 0){
934         ky_min = dv(y - max_y, a / g, true);
935         ky_max = dv(y - min_y, a / g, false);
936      }
937      else{
938         ky_min = dv(y - min_y, a / g, true);
939         ky_max = dv(y - max_y, a / g, false);
940      }
941      if(ky_max < ky_min){ return -INF; }
942
943      ll l = max(kx_min, ky_min), r = min(kx_max, ky_max);
944      if(l > r){ return -INF; }
945      if(b - a >= 0){ return (x + y) + l * (b - a) / g; }
946      else{ return (x + y) + r * (b - a) / g; }
947   }
948   void solve(){
949      ll i, j, n, a, b, t;
950      sl(n), sll(a, b), sl(t);
951      if(t == 1){
952         pl(0ll); nl; return;
953      }
954      int g = __gcd(a, b);
955      if((a / g - 1) * g + b > n - 1){ ps("uh-oh!\n"); return; }
956   //   for(i = 1; i <= t; i++){
957   //      if(i + b > n && i - a <= 0){ ps("")}
958   //   }
959      ll ret = find_all_solution(b, -a, t - 1, 0, INF, 0, INF);
960      if(ret == -INF){ ps("uh-oh!\n"); }
961      else{ pl(ret); nl; }
962   }
963   int main(){
964   //    freopen("input.txt","r",stdin);
965   //    freopen("output.txt", "w", stdout);
966      int cs, ts;
967      si(ts);
968      for0(cs, ts){
969         solve();
970      }
971   }
972
973   /******bellman ford*******/
974   int n, m, dis[110];
975   vector < pair <int, int > > adj[110];
976   void input(){
977      int i, j, u, v, w;
978      for0(i, m){
979         siii(u, v, w); u--, v--;
980         adj[u].push_back(mpr(v, w));
981         adj[v].push_back(mpr(u, -w));
982      }
983   }
984   bool bellman_ford(){
985      int i, j, u, v, w, k;
986      for0(i, n){ dis[i] = INF; } dis[0] = 0;
987      for(i = 0; i < n - 1; i++){
988         for0(j, n){
989            for0(k, adj[j].size()){
990               u = j, v = adj[j][k].first, w = adj[j][k].second;
991               if(dis[u] != INF && dis[u] + w < dis[v]){
992                  dis[v] = dis[u] + w;
993               }
994            }
995         }
996      }
997      for0(i, n){
998         for0(j, adj[i].size()){
999            u = i, v = adj[i][j].first, w = adj[i][j].second;
1000           if(dis[u] != INF && dis[u] + w < dis[v]){
1001              return true;
1002           }
```

```
        }
    }
    return false;
}
void solve(){
    int i, j;
    if(bellman_ford()){ ps("Y"); nl; }
    else{ ps("N\n"); }
    for0(i, n){ adj[i].clear(); }
}
int main(){
//    freopen("input.txt","r",stdin);
//    freopen("output.txt", "w", stdout);
    while(sii(n, m) == 2){
        if(n == 0 && m == 0){ break; }
        input();
        solve();
    }
}


/*****0-1 bfs***************/
int n, m;
char ara[1010][1010];
int dis[1010][1010];
int dx[] = {-1, +0, +1, +0};
int dy[] = {+0, -1, +0, +1};
deque < pair <int, int > > dq;
void input(){
    int i, j;
    sii(n, m);
    for0(i, n){
        ss(ara[i]);
    }
}
bool valid(int x, int y){
    return (x >= 0 && x <= n - 1 && y >= 0 && y <= m - 1);
}
int bfs(){
    int i, j, x, y, go_x, go_y, w;
    for0(i, n){
        for0(j, m){ dis[i][j] = INF; }
    }
    while(!dq.empty()){ dq.pop_back(); }
    dis[0][0] = 0, dq.push_back(mpr(0, 0));
    while(!dq.empty()){
        x = dq.front().first, y = dq.front().second, dq.pop_front();
        if(x == n - 1 && y == m - 1){ return dis[x][y]; }
        for0(i, 4){
            go_x = x + dx[i];
            go_y = y + dy[i];
            if(valid(go_x, go_y)){
                if(ara[x][y] != ara[go_x][go_y]){ w = 1; }
                else{ w = 0; }
                if(dis[x][y] + w < dis[go_x][go_y]){
                    dis[go_x][go_y] = dis[x][y] + w;
                    w == 0 ? dq.push_front(mpr(go_x, go_y)) : dq.push_back(mpr(go_x, go_y));
                }
            }
        }
    }
    return INF;
}
void solve(){
    int i, j;
    int ret = bfs();
    pi(ret); nl;
}
int main(){
//    freopen("input.txt","r",stdin);
//    freopen("output.txt", "w", stdout);
    int cs, ts;
```

```
1075     si(ts);
1076     for0(cs, ts){
1077        input();
1078        printf("Case %d: ", cs + 1);
1079        solve();
1080     }
1081  }
1082
1083
1084  /***shortest tour for every vertex **************/
1085  int n, ht[2010];
1086  int mat[2010][2010];
1087  vector <int> adj[2010];
1088  queue <int> q;
1089  void input(){
1090     int i, j;
1091     si(n);
1092     for0(i, n){
1093        for0(j, n){
1094           si(mat[i][j]);
1095           if(mat[i][j] == 1){
1096              adj[i].push_back(j);
1097           }
1098        }
1099     }
1100  }
1101  int bfs(int src){
1102     int i, j, sol = INF, u, v;
1103     for0(i, n){ ht[i] = -1; }
1104     while(!q.empty()){ q.pop(); }
1105     q.push(src), ht[src] = 0;
1106     while(!q.empty()){
1107        u = q.front(), q.pop();
1108        for0(i, adj[u].size()){
1109           v = adj[u][i];
1110           if(v == src){ return ht[u] + 1; }
1111           else if(ht[v] == -1){
1112              ht[v] = ht[u] + 1;
1113              q.push(v);
1114           }
1115        }
1116     }
1117     return sol;
1118  }
1119  void solve(){
1120     int i, j, sol;
1121     for0(i, n){
1122        sol = INF;
1123        if(adj[i].size()){ sol = bfs(i); }
1124        if(sol == INF){ ps("NO WAY\n"); }
1125        else{ pi(sol); nl; }
1126     }
1127  }
1128  int main(){
1129  //    freopen("input.txt","r",stdin);
1130  //    freopen("output.txt", "w", stdout);
1131
1132     input();
1133     solve();
1134  }
1135
1136
1137  /***************shortest cycle in the graph ******/
1138  int n, m, ht[510], mark[510];
1139  vector <int> adj[510];
1140  queue < int > Q;
1141  void input(){
1142     int i, j;
1143     sii(n, m);
1144     for0(i, m){
1145        int u, v; sii(u, v);
1146        adj[u].push_back(v);
```

```
1147         adj[v].push_back(u);
1148      }
1149   }
1150   int bfs(int src){
1151      int i, j, u, v, ret = INF;
1152      for0(i, n){ ht[i] = INF; }
1153      while(!Q.empty()){ Q.pop(); }
1154      ht[src] = 0; Q.push(src);
1155      while(!Q.empty()){
1156         u = Q.front(), Q.pop();
1157         for0(i, adj[u].size()){
1158            v = adj[u][i];
1159            if(ht[v] == INF){
1160               if(u == src){ mark[v] = v; }
1161               else{ mark[v] = mark[u]; }
1162               ht[v] = ht[u] + 1, Q.push(v);
1163            }
1164            else if(mark[u] != mark[v] && u != src && v != src){ ret = min(ret, ht[u] +
      ht[v] + 1); }
1165         }
1166      }
1167      return ret;
1168   }
1169   void solve(){
1170      int i, j, mn = INF;
1171      for0(i, n){ mn = min(mn, bfs(i)); }
1172      if(mn == INF){ ps("impossible\n"); }
1173      else{ pi(mn); nl; }
1174      for0(i, n){ adj[i].clear(); }
1175   }
1176   int main(){
1177   //    freopen("input.txt","r",stdin);
1178   //    freopen("output.txt", "w", stdout);
1179      int cs, ts;
1180      si(ts);
1181      for0(cs, ts){
1182         input();
1183         printf("Case %d: ", cs + 1);
1184         solve();
1185      }
1186   }
1187
1188
1189   /*******dsu************/
1190   int root[SZ];
1191   struct DSU{
1192      int parent[SZ];
1193      void ini(int n){
1194         int i, j;
1195         for0(i, n){ parent[i] = i; }
1196      }
1197      int get_parent(int u){
1198         if(u == parent[u]){ return u; }
1199         return parent[u] = get_parent(parent[u]);
1200      }
1201      void union_tree(int u, int v){
1202         int pu = get_parent(u);
1203         int pv = get_parent(v);
1204         if(pu == pv){ return; }
1205         int x = rand() % 2;
1206         root[pv] = root[pu];
1207         x == 1 ? (parent[pv] = pu) : (parent[pu] = pv);
1208      }
1209      int get_root(int u){
1210         return root[get_parent(u)];
1211      }
1212   }dsu;
1213   int n, q, sol[SZ], W[SZ];
1214   ll depth[SZ];
1215   vector <int> adj[SZ];
1216   int parent[SZ];
1217   vector < pair <bool, int > > q_list;
```

```cpp
vector < pair <int, int > > act_edge;
bool bad[SZ];
void input(){
   int i, j;
   si(n);
   for1(i, n - 1){
      si(parent[i]);
      adj[--parent[i]].push_back(i);
   }
   for1(i, n - 1){ si(W[i]); }
}
void dfs(int src, int r){
   int i, j, u; root[src] = r;
   for0(i, adj[src].size()){
      u = adj[src][i];
      if(!bad[u]){
         act_edge.push_back(mpr(src, u));
         dfs(u, r);
      }
      else{ dfs(u, u); }
   }
}
void dfs1(int src, ll d){
   int i, j, u; depth[src] = d;
   for0(i, adj[src].size()){
      u = adj[src][i];
      dfs1(u, d + (ll)W[u]);
   }
}
void solve(){
   int i, j, t, u, v;
   dfs1(0, 0);
   si(q);
   for0(i, q){
      sii(t, u); u--;
      q_list.push_back(mpr(t == 1 ? true : false, u));
      if(t == 1){ bad[u] = true; }
   }
   dfs(0, 0);
   dsu.ini(n);
   for0(i, act_edge.size()){
      u = act_edge[i].first, v = act_edge[i].second;
      dsu.union_tree(u, v);
   }
   for(i = q - 1; i >= 0; i--){
      t = q_list[i].first;
      if(t){
         u = parent[q_list[i].second], v = q_list[i].second;
         dsu.union_tree(u, v);
      }
      else{
         u = q_list[i].second;
         sol[i] = dsu.get_root(u);
      }
   }
   for0(i, q){
      if(!q_list[i].first){
         pi(sol[i] + 1); sp; pl(depth[q_list[i].second] - depth[sol[i]]); nl;
      }
   }
}
int main(){
//    freopen("input.txt","r",stdin);
//    freopen("output.txt", "w", stdout);

   input();
   solve();
}


/**********mcmf***************/
namespace mcmf{
```

```
1290        const int MAX = 5 * 100010;
1291        long long cap[MAX], flow[MAX], cost[MAX], dis[MAX];
1292        int n, m, s, t, Q[10000010], adj[MAX], link[MAX], last[MAX], from[MAX], visited[MAX];
1293        void init(int nodes, int source, int sink){
1294            m = 0, n = nodes, s = source, t = sink;
1295            for (int i = 0; i <= n; i++) last[i] = -1;
1296        }
1297        void addEdge(int u, int v, long long c, long long w){
1298            adj[m] = v, cap[m] = c, flow[m] = 0, cost[m] = +w, link[m] = last[u], last[u] =
    m++;
1299            adj[m] = u, cap[m] = 0, flow[m] = 0, cost[m] = -w, link[m] = last[v], last[v] =
    m++;
1300        }
1301        bool spfa(){
1302            int i, j, x, f = 0, l = 0;
1303            for (i = 0; i <= n; i++) visited[i] = 0, dis[i] = INF;
1304            dis[s] = 0, Q[l++] = s;
1305            while (f < l){
1306                i = Q[f++];
1307                for (j = last[i]; j != -1; j = link[j]){
1308                    if (flow[j] < cap[j]){
1309                        x = adj[j];
1310                        if (dis[x] > dis[i] + cost[j]){
1311                            dis[x] = dis[i] + cost[j], from[x] = j;
1312                            if (!visited[x]){
1313                                visited[x] = 1;
1314                                if (f && rand() & 7) Q[--f] = x;
1315                                else Q[l++] = x;
1316                            }
1317                        }
1318                    }
1319                }
1320                visited[i] = 0;
1321            }
1322            return (dis[t] != INF);
1323        }
1324        pair <long long, long long> solve(){
1325            int i, j;
1326            long long mincost = 0, maxflow = 0;
1327            while (spfa()){
1328                long long aug = INF;
1329                for (i = t, j = from[i]; i != s; i = adj[j ^ 1], j = from[i]){
1330                    aug = min(aug, cap[j] - flow[j]);
1331                }
1332                for (i = t, j = from[i]; i != s; i = adj[j ^ 1], j = from[i]){
1333                    flow[j] += aug, flow[j ^ 1] -= aug;
1334                }
1335                maxflow += aug, mincost += aug * dis[t];
1336            }
1337            return make_pair(mincost, maxflow);
1338        }
1339    }
1340    int N, M, Z, T, grid[300][300], p[5010], q[5010], h[5010];
1341    vector <int> vec[300][300];
1342    char str[300][300];
1343    const int dx[] = {-1, +0, +1, +0};
1344    const int dy[] = {+0, -1, +0, +1};
1345    void input(){
1346        int i, j;
1347        sii(N, M), sii(Z, T);
1348        for0(i, N){ ss(str[i]); }
1349        for0(i, N){
1350            for0(j, M){
1351                grid[i][j] = -1;
1352            }
1353        }
1354        for0(i, Z){
1355            int x, y;
1356            sii(x, y), sii(p[i], q[i]), si(h[i]);  --x, --y;
1357            grid[x][y] = i;
1358            vec[x][y].push_back(i);
1359        }
```

```
}
int get_in(int i, int j, int k){
    return i * M * (T + 1) + j * (T + 1) + k + 1;
}
int get_out(int i, int j, int k){
    return N * M * (T + 1) + i * M * (T + 1) + j * (T + 1) + k + 1;
}
bool valid(int x, int y){
    return (x >= 0 && x <= N - 1 && y >= 0 && y <= M - 1 && str[x][y] != '#');
}
void solve(){
    int i, j, idx, x, y, k, l, mx_idx, mx;
    mcmf :: init(2 * N * M * (T + 1) + 2, 0, 2 * N * M * (T + 1) + 1);
    for0(i, N){
        for0(j, M){
            if(str[i][j] == 'S'){ mcmf :: addEdge(mcmf :: s, get_in(i, j, 0), 1, 0); }
            if(str[i][j] != '#'){
                mcmf :: addEdge(get_in(i, j, T), get_out(i, j, T), 1, 0);
                mcmf :: addEdge(get_out(i, j, T), mcmf :: t, 1, 0);
            }
        }
    }
    for0(i, N){
        for0(j, M){
            for0(k, T){
                if(grid[i][j] >= 0){
                    mcmf :: addEdge(get_in(i, j, k), get_out(i, j, k), 1, 0);
                    mx = 0;
                    for0(l, vec[i][j].size()){
                        idx = vec[i][j][l];
                        if(k >= p[idx] && k < q[idx]){
                            mx = max(mx, h[idx]);
                        }
                    }
                    mcmf :: addEdge(get_out(i, j, k), get_in(i, j, k + 1), 1, -mx);
                }
                else if(str[i][j] != '#'){
                    mcmf :: addEdge(get_in(i, j, k), get_out(i, j, k), 1, 0);
                    mcmf :: addEdge(get_out(i, j, k), get_in(i, j, k + 1), 1, 0);
                }
                if(str[i][j] != '#'){
                    for0(l, 4){
                        x = i + dx[l], y = j + dy[l];
                        if(valid(x, y)){
                            mcmf :: addEdge(get_out(i, j, k), get_in(x, y, k + 1), 1, 0);
                        }
                    }
                }
            }
        }
    }
    pair <ll, ll> ret = mcmf :: solve();
    pl(-ret.first); nl;
}
int main(){
//    freopen("input.txt","r",stdin);
//    freopen("output.txt", "w", stdout);
    input();
    solve();
}
```