Team notebook

DU_SwampFire

November 15, 2019

Contents

1 end_pos_set_for_suffix_automata

```
inline 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;
}
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 state{
   int len, link;
   map <char, int> next;
};
state st[SZ * 2];
int to_state = 0, last, f_occ[SZ * 2], d[2 * SZ], sbtr[2 * SZ], vrtx[2 * SZ];
int t = 0;
vector <int> adj[2 * SZ];
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, f_occ[cur] = st[cur].len - 1;
   st[cur].next.clear();
   int p = last;
   while(p != -1 && st[p].next.find(c) == st[p].next.end()){
```

```
30       st[p].next[c] = cur;
31       p = st[p].link;
32    }
33    if(p == -1){ st[cur].link = 0; }
34    else{
35       int q = st[p].next[c];
36       if(st[p].len + 1 == st[q].len){ st[cur].link = q; }
37       else{
38          int clone = to_state++;
39          st[clone].len = st[p].len + 1;
40          st[clone].next = st[q].next;
41          st[clone].link = st[q].link;
42          f_occ[clone] = f_occ[q];
43          while(p != -1 && st[p].next[c] == q){
44             st[p].next[c] = clone;
45             p = st[p].link;
46          }
47          st[q].link = st[cur].link = clone;
48       }
49    } last = cur;
50 }
51 int n; ll mx_up, mx_dn;
52 char str[SZ];
53 void build(){
54    int i, j, u, v;
55    sa_init();
56    for0(i, n){ sa_extend(str[i]); }
57    for0(i, to_state){
58       u = i, v = st[i].link;
59       if(v != -1){ adj[v].push_back(u); }
60    }
61 }
62 void input(){
63    int i, j;
64    ss(str), n = strlen(str);
65 }
66 void dfs(int src){
67    int i, j, u;
68    vrtx[t++] = src, d[src] = t - 1, sbtr[src] = 1;
69    for0(i, adj[src].size()){
70       u = adj[src][i];
71       dfs(u), sbtr[src] += sbtr[u];
72    }
73 }
74 set <int> end_pos;
75 inline void update_max(ll up, ll dn){
76    ll g = __gcd(up, dn); up /= g, dn /= g;
77    if(mx_up * dn <= up * mx_dn){
78       mx_up = up, mx_dn = dn;
79    }
80 }
81 inline void update(int src, int pos){
82    ll up, dn;
83    if(end_pos.empty()){ return; }
84    std :: set <int> :: iterator it;
85    it = end_pos.upper_bound(pos);
86    if(it == end_pos.end()){
87       --it;
88       if(abs(pos - *it) >= st[src].len){
89          up = (ll)abs(pos - *it) + (ll)st[src].len;
90          dn = (ll)abs(pos - *it);
```

```
91          update_max((ll)up, (ll)dn);
92       }
93       return;
94    }
95    if(*it - pos >= st[src].len){
96       up = (ll)abs(*it - pos) + (ll)st[src].len;
97       dn = (ll)abs(*it - pos);
98       update_max((ll)up, (ll)dn);
99    }
100   if(it != end_pos.begin()){
101      --it;
102      if(abs(pos - *it) >= st[src].len){
103         up = (ll)abs(pos - *it) + (ll)st[src].len;
104         dn = (ll)abs(pos - *it);
105         update_max((ll)up, (ll)dn);
106      }
107   }
108 }
109 void dsu(int src, bool keep){
110    int i, j, mx = -1, bg = -1, u, v;
111    for0(i, adj[src].size()){
112       u = adj[src][i];
113       if(sbtr[u] > mx){
114          mx = sbtr[u], bg = u;
115       }
116    }
117    for0(i, adj[src].size()){
118       u = adj[src][i];
119       if(u != bg){
120          dsu(u, false);
121       }
122    }
123    if(bg != -1){ dsu(bg, true); }
124    for0(i, adj[src].size()){
125       u = adj[src][i];
126       if(u != bg){
127          for(j = d[u]; j <= d[u] + sbtr[u] - 1; j++){
128             v = vrtx[j];
129             if(f_occ[v] + 1 - st[v].len == 0){
130                update(src, f_occ[v]);
131                end_pos.insert(f_occ[v]);
132             }
133          }
134       }
135    }
136    if(f_occ[src] + 1 - st[src].len == 0){
137       update(src, f_occ[src]), end_pos.insert(f_occ[src]);
138    }
139 //  std :: set <int> :: iterator it;
140 //  cout << src << "-->"; nl;
141 //  for(it = end_pos.begin(); it != end_pos.end(); ++it){
142 //     pi(*it); sp;
143 //  }nl;
144    if(!keep){ end_pos.clear(); }
145 }
146 int h[2][SZ], base[] = {47, 31}, P[2][SZ], mod[] = {MOD, MOD + 2};
147 void make_hash(){
148    int i, j;
149    for0(i, 2){
150       for(j = 1, P[i][0] = 1; j <= SZ - 2; j++){
151          P[i][j] = mul(P[i][j - 1], base[i], mod[i]);
```

```
152         }
153      }
154      for0(i, 2){
155         for(j = n - 1, h[i][n] = 0; j >= 0; j--){
156            h[i][j] = add(str[j] - 'a' + 1, mul(base[i], h[i][j + 1], mod[i]), mod[i]);
157         }
158      }
159   }
160   inline int get_hash(int l, int r, int idx){
161      return add(h[idx][l], -mul(h[idx][r + 1], P[idx][r - l + 1], mod[idx]), mod[idx]);
162   }
163   int lcp(int l1, int r1, int l2, int r2){
164      if(l1 > r1 || l2 > r2){ return 0; }
165      int lo = 0, hi = min(r1 - l1 + 1, r2 - l2 + 1), mid;
166      int f0, f1, g0, g1;
167      while(lo < hi){
168         mid = (lo + hi + 1) >> 1;
169         f0 = get_hash(l1, l1 + mid - 1, 0);
170         f1 = get_hash(l1, l1 + mid - 1, 1);
171         g0 = get_hash(l2, l2 + mid - 1, 0);
172         g1 = get_hash(l2, l2 + mid - 1, 1);
173         if(f0 == g0 && f1 == g1){ lo = mid; }
174         else{ hi = mid - 1; }
175      }
176      return lo;
177   }
178   int lcs(int l1, int r1, int l2, int r2){
179      if(l1 > r1 && l2 > r2){ return 0; }
180      int lo = 0, hi = min(r1 - l1 + 1, r2 - l2 + 1), mid;
181      int f0, f1, g0, g1;
182      while(lo < hi){
183         mid = (lo + hi + 1) >> 1;
184         f0 = get_hash(r1 - mid + 1, r1, 0);
185         f1 = get_hash(r1 - mid + 1, r1, 1);
186         g0 = get_hash(r2 - mid + 1, r2, 0);
187         g1 = get_hash(r2 - mid + 1, r2, 1);
188         if(f0 == g0 && f1 == g1){ lo = mid; }
189         else{ hi = mid - 1; }
190      }
191      return lo;
192   }
193   pair <int, int > vec[SZ];
194   void solve(){
195      int i, j, cnt, k;
196      build();
197      t = 0, dfs(0);
198      mx_up = -INF, mx_dn = 1; dsu(0, false);
199      make_hash();
200      for(i = 1; i <= n; i++){
201         for(j = 0, cnt = 0; j + i - 1 < n; j += i){
202            int h0 = get_hash(j, j + i - 1, 0);
203            int h1 = get_hash(j, j + i - 1, 1);
204            vec[cnt++] = mpr(h0, h1);
205         }
206         for(j = 0; j < cnt; j = k){
207            for(k = j; k < cnt && vec[k] == vec[j]; k++){}
208            int x = lcs(0, j * i - 1, j * i, j * i + i - 1);
209            int y = lcp(k * i, n - 1, j * i, j * i + i - 1);
210            ll up = (ll)x + (ll)y + (ll)(k - j) * (ll)i;
211            ll dn = i;
212            update_max(up, dn);
```

```
213         }
214     }
215     printf("%lld/%lld\n", mx_up, mx_dn);
216 }
217 int main(){
218 //   freopen("input.txt","r",stdin);
219 //   freopen("output.txt", "w", stdout);
220 
221     input();
222     solve();
223 }
```

2 expected_number_of_components

```
1  //First we consider how to calculate E(X).
2  //
3  //The number of connected components equals to the number of nodes
4  //minus the number of edges and then add the number of rings in it.
5  //So we can calculate the possibility of removing one node,
6  //one edge or one single ring.
7  //
8  //Then we can split the variance, it is equals to E(X2)2E(X)2+E(X)2=E(X2)E(X)2.
9  //Then we can again to split X2.
10 //Let the number nodes equal to a, the number edges equal to b,
11 //the number rings equal to c.
12 //Then X2=(ab+c)2=a2+b2+c22ab2bc+2ac.
13 //We can find there is contribution between a pair of nodes, edges, rings
14 //(the two may be the same) and between a node and an edge,
15 //a node and a ring, an edge and a ring. Then we can calculate the possibility of
16 //such pair that the elements in it remains at the same time.
17 //The answer is the same when the pair is a ring and a node on it,
18 //or when it is a ring and a node not on it,
19 //or an edge with one of its end point ...
20 //If we consider all the situation of intersection and not intersection,
21 //we can get a liner algorithm. But the Time Complexity is O(nlogn)
22 //since we need to calculate the multiplicative inverse of modulo.
23 int big_mod(int v, int p){
24     if(p == 0){ return 1; }
25     int ret = big_mod(v, p / 2);
26     if(p % 2 == 0){ return mul(ret, ret); }
27     return mul(ret, mul(ret, v));
28 }
29 int n, m, pwr[2 * SZ], inv_pwr[2 * SZ];
30 vector <int> adj[SZ];
31 bool vis[SZ], is_art[SZ];
32 int Time, low[SZ], dis[SZ], bcc_cnt;
33 vector <int> bcc[SZ], art_vec[SZ];
34 stack <int> S;
35 void pop_bcc(int s, int u){
36     is_art[s] = true;
37     bcc[bcc_cnt].push_back(s);
38     while(true){
39         bcc[bcc_cnt].push_back(S.top());
40         if(S.top() == u){ S.pop(); break; }
41         S.pop();
42     }
43     bcc_cnt++;
44     if(bcc[bcc_cnt - 1].size() <= 2){
```

```
45        bcc[bcc_cnt - 1].clear();
46        bcc_cnt--;
47     }
48  }
49  void find_bcc(int src, int par){
50     S.push(src);
51     int i, u, child = 0; vis[src] = true, Time++, dis[src] = low[src] = Time;
52     for0(i, adj[src].size()){
53        u = adj[src][i];
54        if(!vis[u]){
55           child++, find_bcc(u, src);
56           low[src] = min(low[src], low[u]);
57           if(par != -1 && low[u] >= dis[src]){ pop_bcc(src, u); }
58           else if(par == -1){
59              if(child > 1){ pop_bcc(src, u); }
60           }
61        } else if(par != u){ low[src] = min(low[src], dis[u]); }
62     } if(par == -1 && child > 1){ is_art[src] = true; }
63  }
64  void process_bcc(){
65     int i, j;
66     bool f;
67     bcc_cnt = 0;
68     for0(i, n){
69        if(!vis[i]){
70           Time = 0;
71           find_bcc(i, -1);
72           f = false;
73           while(!S.empty()){
74              f = true;
75              bcc[bcc_cnt].push_back(S.top());
76              S.pop();
77           } if(f){
78              bcc_cnt++;
79              if(bcc[bcc_cnt - 1].size() <= 2){
80                 bcc[bcc_cnt - 1].clear();
81                 bcc_cnt--;
82              }
83           }
84        }
85     }
86  }
87  void input(){
88     int i, j;
89     sii(n, m);
90     for0(i, m){
91        int u, v;
92        sii(u, v), u--, v--;
93        adj[u].push_back(v), adj[v].push_back(u);
94     }
95  }
96  inline int comb(int a){
97     return mul(a, a - 1);
98  }
99  void solve(){
100    int i, j, sol = 0, sz, baki = 0, sum = 0, sz1;
101    process_bcc();
102    //a
103    int a = mul(n, inv_pwr[1]);
104    //b
105    int b = mul(m, inv_pwr[2]);
```

```
    //c
    int c = 0;
    for0(i, bcc_cnt){
       sz = bcc[i].size();
       c = add(c, inv_pwr[sz]);
    }
    //a^2;
    int a_2 = mul(comb(n), inv_pwr[2]);
    a_2 = add(a_2, mul(n, inv_pwr[1]));
    //b^2
    int b_2 = 0;
    for0(i, n){
       sz = adj[i].size();
       b_2 = add(b_2, mul(comb(sz), inv_pwr[3]));
       b_2 = add(b_2, -mul(comb(sz), inv_pwr[4]));
    }
    b_2 = add(b_2, mul(comb(m), inv_pwr[4]));
    b_2 = add(b_2, mul(m, inv_pwr[2]));
    //c^2
    int c_2 = 0;
    for0(i, bcc_cnt){
       for0(j, bcc[i].size()){
          if(is_art[bcc[i][j]]){
             art_vec[bcc[i][j]].push_back(i);
          }
       }
    }
    for(i = 0; i < n; i++){
       for(j = 0, sum = 0; j < art_vec[i].size(); j++){
          sz = bcc[art_vec[i][j]].size();
          sum = add(sum, inv_pwr[sz]);
       }
       for0(j, art_vec[i].size()){
          sz = bcc[art_vec[i][j]].size();
          c_2 = add(c_2, mul(add(sum, -inv_pwr[sz]), inv_pwr[sz - 1]));
          c_2 = add(c_2, -mul(add(sum, -inv_pwr[sz]), inv_pwr[sz]));
       }
    }
    for(i = 0, sum = 0; i < bcc_cnt; i++){
       sz = bcc[i].size();
       sum = add(sum, inv_pwr[sz]);
    }
    for0(i, bcc_cnt){
       sz = bcc[i].size();
       c_2 = add(c_2, mul(add(sum, -inv_pwr[sz]), inv_pwr[sz]));
       c_2 = add(c_2, inv_pwr[sz]);
    }
    //ab
    int ab = 0;
    ab = add(ab, mul(mul(2, m), inv_pwr[2]));
    ab = add(ab, mul(mul(m, n - 2), inv_pwr[3]));
    //bc
    int bc = 0;
    for0(i, bcc_cnt){
       sz = bcc[i].size();
       bc = add(bc, mul(sz, inv_pwr[sz]));
       bc = add(bc, -mul(sz, inv_pwr[sz + 2]));
       bc = add(bc, mul(m, inv_pwr[sz + 2]));
       for0(j, bcc[i].size()){
          sz1 = adj[bcc[i][j]].size();
          sz1 = add(sz1, -2);
```

```
            bc = add(bc, mul(sz1, inv_pwr[sz + 1]));
            bc = add(bc, -mul(sz1, inv_pwr[sz + 2]));
        }
    }
    //ac
    int ca = 0;
    for0(i, bcc_cnt){
        sz = bcc[i].size();
        ca = add(ca, mul(sz, inv_pwr[sz]));
        ca = add(ca, -mul(sz, inv_pwr[sz + 1]));
        ca = add(ca, mul(n, inv_pwr[sz + 1]));
    }
    int e_x = add(a, add(-b, c));
    sol = 0;
    sol = add(sol, -mul(e_x, e_x));
    sol = add(sol, add(a_2, add(b_2, c_2)));
    sol = add(sol, -mul(2, ab));
    sol = add(sol, -mul(2, bc));
    sol = add(sol, mul(2, ca));
    pi(sol); nl;
}
void pre_process(){
    int i, j, inv_2 = big_mod(2, MOD - 2);
    for(i = 1, pwr[0] = 1, inv_pwr[0] = 1; i < 2 * SZ - 2; i++){
        pwr[i] = mul(pwr[i - 1], 2);
        inv_pwr[i] = mul(inv_pwr[i - 1], inv_2);
    }
}
int main(){
//   freopen("input.txt","r",stdin);
//   freopen("output.txt", "w", stdout);
    pre_process();
    input();
    solve();
}
```

3 function_decomposition

```
int add(int _a, int _b){
    _a = (_a + MOD) % MOD;
    _b = (_b + MOD) % MOD;
    return (_a + _b) % MOD;
}
int mul(int _a, int _b){
    _a = (_a + MOD) % MOD;
    _b = (_b + MOD) % MOD;
    return ((ll)((ll)_a * (ll)_b)) % MOD;
}
int n, ara[SZ], sccCnt = 0, sccRoot[SZ], sccNo[SZ], rootNo[SZ], sbtr[SZ], ht[SZ], q,
    ↪ global[SZ], position[SZ], sol[SZ];
const int thresh = 200005;
bool inCycle[SZ];
vector <int> adj[SZ], rev[SZ], scc[SZ], cum[SZ];
vector <int> topo, treeRoot, qlist[SZ], vrtx;
pair <ll, int> query[SZ];
bool vis[SZ];
void input(){
    int i, j;
```

```
   si(n);
   for1(i, n){
      si(ara[i]);
      if(ara[i] == i){ inCycle[ara[i]] = true; }
      adj[ara[i]].push_back(i); rev[i].push_back(ara[i]);
   }
//  ppvec(i, j, n + 1, adj);
}
void dfs0(int src){
   int i, j; vis[src] = true;
   for0(i, adj[src].size()){
      int u = adj[src][i];
      if(!vis[u]){ dfs0(u); }
   } topo.push_back(src);
}
void dfs1(int src){
   int i, j; vis[src] = true; scc[sccCnt].push_back(src);
   sccNo[src] = sccCnt;
   for0(i, rev[src].size()){
      int u = rev[src][i];
      if(!vis[u]){ dfs1(u); }
   }
}
void dfs2(int src, int rt){
   int i, j; vis[src] = true; sbtr[src] = 1;
   rootNo[src] = rt;
   vrtx.push_back(src);
   for0(i, adj[src].size()){
      int u = adj[src][i];
      if(!vis[u]){ dfs2(u, rt), sbtr[src] += sbtr[u]; }
   }
}
void dfs3(int src, int rt, int d, int len, int idx){
   int i, j; vis[src] = true; ht[src] = d;
   cum[idx][ht[src]]++;
   for0(i, adj[src].size()){
      int u = adj[src][i];
      if(!vis[u]){
         if(inCycle[u]){ dfs3(u, u, d + 1, len, idx); }
         else{ dfs3(u, rt, d + 1, len, idx); }
      }
   }
}

void dsu(int src, int par, bool keep){
   int i, j, u, bg = -1, mx = -1, v;
   ll d;
   for0(i, adj[src].size()){
      u = adj[src][i];
      if(u != par && u != rootNo[src]){
         if(sbtr[u] > mx){ mx = sbtr[u], bg = u; }
      }
   }
   for0(i, adj[src].size()){
      u = adj[src][i];
      if(u != par && u != rootNo[src] && u != bg){
         dsu(u, src, false);
      }
   }
   if(mx != -1){ dsu(bg, src, true); }
   global[ht[src]]++;
```

```
    for0(i, adj[src].size()){
       u = adj[src][i];
       if(u != par && u != rootNo[src] && u != bg){
          for(j = position[u]; j <= position[u] + sbtr[u] - 1; j++){
             v = vrtx[j];
             global[ht[v]]++;
          }
       }
    }
    for0(i, qlist[src].size()){
       int idx = qlist[src][i];
       d = query[idx].first;
       if((ll)ht[src] + d <= (ll)thresh){
          sol[idx] += global[(ll)ht[src] + d];
       }
    }
    if(!keep){
       for(j = position[src]; j <= position[src] + sbtr[src] - 1; j++){
          v = vrtx[j];
          global[ht[v]]--;
       }
    }
}
void solve(){
    int i, j, u, len, r, dis, sz;
    ll d;
    for(i = 1; i <= n; i++){
       if(!vis[i]){ dfs0(i); }
    }
//  pvec(i, topo);
    reverse(topo.begin(), topo.end());
    for(i = 1; i <= n; i++){ vis[i] = false; }
    for(i = 0, sccCnt = 0; i < topo.size(); i++){
       if(!vis[topo[i]]){
          dfs1(topo[i]);
          sccRoot[sccCnt] = topo[i];
          sccCnt++;
       }
    }
    for0(i, sccCnt){
       if(scc[i].size() >= 2){
          for0(j, scc[i].size()){
             inCycle[scc[i][j]] = true;
          }
       }
    }
    for(i = 1; i <= n; i++){ vis[i] = false; }
    for(i = 0; i < sccCnt; i++){
       if(scc[i].size() >= 2 || (scc[i].size() == 1 && inCycle[scc[i][0]])){
          treeRoot.push_back(sccRoot[i]);
          dfs2(sccRoot[i], sccRoot[i]);
       }
       else if(scc[i].size() == 1 && rev[scc[i][0]].size() == 0){
          treeRoot.push_back(scc[i][0]);
          dfs2(scc[i][0], scc[i][0]);
       }
    }
    for0(i, vrtx.size()){
       u = vrtx[i], position[u] = i;
    }
    for(i = 1; i <= n; i++){ vis[i] = false; }
```

```
   for(i = 0; i < treeRoot.size(); i++){
      u = treeRoot[i];
      if(inCycle[u]){
         cum[u].resize(sbtr[u] + 1);
         len = scc[sccNo[u]].size();
         dfs3(u, u, 0, len, u);
         for(j = 0; j < cum[u].size(); j++){
            cum[u][j] = j - len < 0 ? cum[u][j] : cum[u][j] + cum[u][j - len];
         }
      }
   }
   si(q);
   for0(i, q){
      scanf("%lld %d", &query[i].first, &query[i].second);
      qlist[query[i].second].push_back(i);
      u = query[i].second; d = query[i].first;
//      pi(u); nl;
      if(inCycle[u]){
         r = rootNo[u];
         len = scc[sccNo[r]].size();
         dis = len - ht[u];
         d -= (ll)dis; sz = cum[r].size();
         if(d >= sz){
            d = d - ((d - (ll)sz + (ll)len) / (ll)len) * (ll)len;
         }
         if(d < 0){ continue; }
         sol[i] += cum[r][d];
      }
   }
   for0(i, treeRoot.size()){
      u = treeRoot[i];
      dsu(u, -1, false);
   }
   for0(i, q){
      pi(sol[i]); nl;
   }
}
int main(){
//   freopen("input.txt","r",stdin);
//   freopen("output.txt", "w", stdout);
   input();
   solve();
}
```

4 iterative_segment_tree

```
const int N = 1e5; // limit for array size
int n; // array size
int t[2 * N];

void build() { // build the tree
  for (int i = n - 1; i > 0; --i) t[i] = t[i<<1] + t[i<<1|1];
}

void modify(int p, int value) { // set value at position p
  for (t[p += n] = value; p > 1; p >>= 1) t[p>>1] = t[p] + t[p^1];
}

```

```
int query(int l, int r) { // sum on interval [l, r)
  int res = 0;
  for (l += n, r += n; l < r; l >>= 1, r >>= 1) {
    if (l&1) res += t[l++];
    if (r&1) res += t[--r];
  }
  return res;
}

int main() {
  scanf("%d", &n);
  for (int i = 0; i < n; ++i) scanf("%d", t + n + i);
  build();
  modify(0, 1);
  printf("%d\n", query(3, 11));
  return 0;
}
```

5 jumping_frog_with_update

```
int n, ara[SZ], q, nxt[SZ], hole[SZ], inside[SZ];
int sqn;
void input(){
   int i, j;
   sii(n, q);
   for0(i, n){si(ara[i]);}
   sqn = sqrt(n) + 1;
}
void solve(){
   int i, j, ret = 0, last;
   for(i = 0; i <= n / sqn; i++){
      for(j = min(n - 1, (i + 1) * sqn - 1); j >= i * sqn; j--){
         if(j + ara[j] > min(n - 1, (i + 1) * sqn - 1)){
            nxt[j] = j + ara[j];
            inside[j] = j;
            hole[j] = 1;
         }
         else{
            nxt[j] = nxt[j + ara[j]];
            inside[j] = inside[j + ara[j]];
            hole[j] = 1 + hole[j + ara[j]];
         }
      }
   }
   for0(i, q){
      int t;
      si(t);
      if(t == 0){
         int idx, v;
         sii(idx, v); idx--;
         ara[idx] = v;
         for(j = min(n - 1, ((idx / sqn) + 1) * sqn - 1); j >= (idx / sqn) * sqn; j--){
            if(j + ara[j] > min(n - 1, ((idx / sqn) + 1) * sqn - 1)){
               inside[j] = j;
               nxt[j] = j + ara[j];
               hole[j] = 1;
            }
            else{
```

```
                nxt[j] = nxt[j + ara[j]];
                inside[j] = inside[j + ara[j]];
                hole[j] = 1 + hole[j + ara[j]];
            }
         }
      }
      else{
         int idx;
         si(idx); idx--;
         for(ret = 0 ;idx < n;){
            last = inside[idx];
            ret += hole[idx];
            idx = nxt[idx];
         }
         pii(last + 1, ret);NL;
      }
   }
}
int main(){
//   freopen("input.txt","r",stdin);
   input();
   solve();
}
```

6 mo_on_tree

```
inline 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;
}
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 state{
   int len, link;
   map <char, int> next;
};
state st[SZ * 2];
int to_state = 0, last, f_occ[SZ * 2], d[2 * SZ], sbtr[2 * SZ], vrtx[2 * SZ];
int t = 0;
vector <int> adj[2 * SZ];
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, f_occ[cur] = st[cur].len - 1;
   st[cur].next.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;
   }
```

```
33     if(p == -1){ st[cur].link = 0; }
34     else{
35        int q = st[p].next[c];
36        if(st[p].len + 1 == st[q].len){ st[cur].link = q; }
37        else{
38           int clone = to_state++;
39           st[clone].len = st[p].len + 1;
40           st[clone].next = st[q].next;
41           st[clone].link = st[q].link;
42           f_occ[clone] = f_occ[q];
43           while(p != -1 && st[p].next[c] == q){
44              st[p].next[c] = clone;
45              p = st[p].link;
46           }
47           st[q].link = st[cur].link = clone;
48        }
49     } last = cur;
50  }
51  int n; ll mx_up, mx_dn;
52  char str[SZ];
53  void build(){
54     int i, j, u, v;
55     sa_init();
56     for0(i, n){ sa_extend(str[i]); }
57     for0(i, to_state){
58        u = i, v = st[i].link;
59        if(v != -1){ adj[v].push_back(u); }
60     }
61  }
62  void input(){
63     int i, j;
64     ss(str), n = strlen(str);
65  }
66  void dfs(int src){
67     int i, j, u;
68     vrtx[t++] = src, d[src] = t - 1, sbtr[src] = 1;
69     for0(i, adj[src].size()){
70        u = adj[src][i];
71        dfs(u), sbtr[src] += sbtr[u];
72     }
73  }
74  set <int> end_pos;
75  inline void update_max(ll up, ll dn){
76     ll g = __gcd(up, dn); up /= g, dn /= g;
77     if(mx_up * dn <= up * mx_dn){
78        mx_up = up, mx_dn = dn;
79     }
80  }
81  inline void update(int src, int pos){
82     ll up, dn;
83     if(end_pos.empty()){ return; }
84     std :: set <int> :: iterator it;
85     it = end_pos.upper_bound(pos);
86     if(it == end_pos.end()){
87        --it;
88        if(abs(pos - *it) >= st[src].len){
89           up = (ll)abs(pos - *it) + (ll)st[src].len;
90           dn = (ll)abs(pos - *it);
91           update_max((ll)up, (ll)dn);
92        }
93        return;
```

```
    }
    if(*it - pos >= st[src].len){
       up = (ll)abs(*it - pos) + (ll)st[src].len;
       dn = (ll)abs(*it - pos);
       update_max((ll)up, (ll)dn);
    }
    if(it != end_pos.begin()){
       --it;
       if(abs(pos - *it) >= st[src].len){
          up = (ll)abs(pos - *it) + (ll)st[src].len;
          dn = (ll)abs(pos - *it);
          update_max((ll)up, (ll)dn);
       }
    }
}
void dsu(int src, bool keep){
    int i, j, mx = -1, bg = -1, u, v;
    for0(i, adj[src].size()){
       u = adj[src][i];
       if(sbtr[u] > mx){
          mx = sbtr[u], bg = u;
       }
    }
    for0(i, adj[src].size()){
       u = adj[src][i];
       if(u != bg){
          dsu(u, false);
       }
    }
    if(bg != -1){ dsu(bg, true); }
    for0(i, adj[src].size()){
       u = adj[src][i];
       if(u != bg){
          for(j = d[u]; j <= d[u] + sbtr[u] - 1; j++){
             v = vrtx[j];
             if(f_occ[v] + 1 - st[v].len == 0){
                update(src, f_occ[v]);
                end_pos.insert(f_occ[v]);
             }
          }
       }
    }
    if(f_occ[src] + 1 - st[src].len == 0){
       update(src, f_occ[src]), end_pos.insert(f_occ[src]);
    }
//  std :: set <int> :: iterator it;
//  cout << src << "-->"; nl;
//  for(it = end_pos.begin(); it != end_pos.end(); ++it){
//     pi(*it); sp;
//  }nl;
    if(!keep){ end_pos.clear(); }
}
int h[2][SZ], base[] = {47, 31}, P[2][SZ], mod[] = {MOD, MOD + 2};
void make_hash(){
    int i, j;
    for0(i, 2){
       for(j = 1, P[i][0] = 1; j <= SZ - 2; j++){
          P[i][j] = mul(P[i][j - 1], base[i], mod[i]);
       }
    }
    for0(i, 2){
```

```
155         for(j = n - 1, h[i][n] = 0; j >= 0; j--){
156             h[i][j] = add(str[j] - 'a' + 1, mul(base[i], h[i][j + 1], mod[i]), mod[i]);
157         }
158     }
159 }
160 inline int get_hash(int l, int r, int idx){
161     return add(h[idx][l], -mul(h[idx][r + 1], P[idx][r - l + 1], mod[idx]), mod[idx]);
162 }
163 int lcp(int l1, int r1, int l2, int r2){
164     if(l1 > r1 || l2 > r2){ return 0; }
165     int lo = 0, hi = min(r1 - l1 + 1, r2 - l2 + 1), mid;
166     int f0, f1, g0, g1;
167     while(lo < hi){
168         mid = (lo + hi + 1) >> 1;
169         f0 = get_hash(l1, l1 + mid - 1, 0);
170         f1 = get_hash(l1, l1 + mid - 1, 1);
171         g0 = get_hash(l2, l2 + mid - 1, 0);
172         g1 = get_hash(l2, l2 + mid - 1, 1);
173         if(f0 == g0 && f1 == g1){ lo = mid; }
174         else{ hi = mid - 1; }
175     }
176     return lo;
177 }
178 int lcs(int l1, int r1, int l2, int r2){
179     if(l1 > r1 && l2 > r2){ return 0; }
180     int lo = 0, hi = min(r1 - l1 + 1, r2 - l2 + 1), mid;
181     int f0, f1, g0, g1;
182     while(lo < hi){
183         mid = (lo + hi + 1) >> 1;
184         f0 = get_hash(r1 - mid + 1, r1, 0);
185         f1 = get_hash(r1 - mid + 1, r1, 1);
186         g0 = get_hash(r2 - mid + 1, r2, 0);
187         g1 = get_hash(r2 - mid + 1, r2, 1);
188         if(f0 == g0 && f1 == g1){ lo = mid; }
189         else{ hi = mid - 1; }
190     }
191     return lo;
192 }
193 pair <int, int > vec[SZ];
194 void solve(){
195     int i, j, cnt, k;
196     build();
197     t = 0, dfs(0);
198     mx_up = -INF, mx_dn = 1; dsu(0, false);
199     make_hash();
200     for(i = 1; i <= n; i++){
201         for(j = 0, cnt = 0; j + i - 1 < n; j += i){
202             int h0 = get_hash(j, j + i - 1, 0);
203             int h1 = get_hash(j, j + i - 1, 1);
204             vec[cnt++] = mpr(h0, h1);
205         }
206         for(j = 0; j < cnt; j = k){
207             for(k = j; k < cnt && vec[k] == vec[j]; k++){}
208             int x = lcs(0, j * i - 1, j * i, j * i + i - 1);
209             int y = lcp(k * i, n - 1, j * i, j * i + i - 1);
210             ll up = (ll)x + (ll)y + (ll)(k - j) * (ll)i;
211             ll dn = i;
212             update_max(up, dn);
213         }
214     }
215     printf("%lld/%lld\n", mx_up, mx_dn);
```

```
}
int main(){
//   freopen("input.txt","r",stdin);
//   freopen("output.txt", "w", stdout);

   input();
   solve();
}
```

7 online_fft

```
int big_mod(int v, int p){
   if(p == 0){ return 1; }
   int ret = big_mod(v, p / 2);
   if(p % 2 == 0){ return mul(ret, ret); }
   else{ return mul(ret, mul(ret, v)); }
}
const int N = (1 << 18) + 10, mod = 998244353, g = 3;
int rev[N], w[N], inv_n;
void prepare(int &n){
   int sz = 31 - __builtin_clz(n); sz = abs(sz);
   int r = big_mod(g, (mod - 1) / n);
   inv_n = big_mod(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; } }
}
int f_a[N], f_b[N];
vector <int> multiply(vector <int> a, vector <int> b){
   int sz = 1, na = (int)(a.size()), nb = (int)(b.size());
   while(sz < na + nb - 1){ sz <<= 1; }
   prepare(sz);
   for(int i = 0; i < sz; ++i){ f_a[i] = i < na ? a[i] : 0; }
   for(int i = 0; i < sz; ++i){ f_b[i] = i < nb ? b[i] : 0; }
   ntt(f_a, sz , 0), ntt(f_b, sz, 0);
   for(int i = 0; i < sz; ++i){ f_a[i] = (ll)f_a[i] * f_b[i] % mod; }
   ntt(f_a, sz, 1); return vector <int> (f_a, f_a + sz);
}
int n, G[SZ], F[SZ], dp[SZ];
bool vis[SZ];
void input(){
   int i, j;
   si(n);
```

```
48    for0(i, n){ si(G[i]); }
49 }
50 vector <int> segmentation(int l1, int r1, int l2, int r2){
51    int i, j;
52    vector <int> a, b;
53    a.resize(r1 - l1 + 1); b.resize(r2 - l2 + 1);
54    for(i = l1, j = 0; i <= n && i <= r1; i++, j++){
55       a[j] = F[i];
56    }
57    for(i = l2, j = 0; i <= n && i <= r2; i++, j++){
58       b[j] = G[i];
59    }
60    return multiply(a, b);
61 }
62 void contribute(int offst, vector <int> &poly){
63    int i, j;
64    for(j = 0; j < poly.size() && j + offst <= n; j++){
65       F[j + offst] = add(F[j + offst], poly[j]);
66    }
67 }
68 int brute(int x){
69    if(x == 0){ return 1; }
70    if(vis[x]){ return dp[x]; }
71    vis[x] = true;
72    int ret = G[x], i;
73    for(i = 0; i <= x - 1; i++){
74       ret = add(ret, mul(brute(i), G[x - i]));
75    }
76    return dp[x] = ret;
77 }
78 void solve(){
79    int i, j, l, ret;
80    F[0] = 1;
81    for(i = 0; i <= n; i++){ F[i] = G[i]; }
82    for(i = 0; i <= n; i++){
83       for(j = 0; (1 << j) <= i + 1; j++){
84          if((i + 1) % (1 << j) == 0){
85             vector <int> a = segmentation(i - (1 << j) + 1, i, 1 << j, (1 << j) + (1 <<
                  ↪ j) - 1);
86             int offst = (1 << j) * (i / (1 << j) + 1);
87             contribute(offst, a);
88             cout << "i-->" << i << " " << "j-->" << j << " " << offst; nl;
89             para(l, 0, n, F);
90          }
91       }
92    } nl;
93    dp[0] = 1;
94    for(i = 0; i <= n; i++){
95       ret = brute(i);
96    }
97    para(i, 0, n, dp);
98    para(i, 0, n, F);
99 }
100 int main(){
101    input();
102    solve();
103 }
```

8 sum_of_kth_power_of_all_possible_subgraph_in_the_tree

```cpp
int add(int _a, int _b){
   _a = (_a + MOD) % MOD;
   _b = (_b + MOD) % MOD;
   return (_a + _b) % MOD;
}
int mul(int _a, int _b){
   _a = (_a + MOD) % MOD;
   _b = (_b + MOD) % MOD;
   return ((ll)((ll)_a * (ll)_b)) % MOD;
}
int n, k, sbtr[SZ], sum[210], dif[SZ][210], sol = 0, fac[SZ], inv[SZ], h[SZ][210],
   ↪ f[210][210];
vector <int> adj[SZ];
void input(){
   int i, j;
   sii(n, k);
   for(i = 0; i < n - 1; i++){
      int u, v; sii(u, v), u--, v--;
      adj[u].push_back(v), adj[v].push_back(u);
   }
}
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 factorial(){
   int i, j;
   for(i = 1, fac[0] = inv[0] = 1; i < SZ - 1; i++){
      fac[i] = mul(fac[i - 1], i);
      inv[i] = mul(inv[i - 1], bigMod(i, MOD - 2));
   }
}
void H(int src){
   int sz = adj[src].size(), i, j, l, u, x;
   for(i = sz - 1; i >= 0; i--){
      for(u = adj[src][i], j = 0; j <= min(k, sbtr[src] - 1); j++){
         if(i == sz - 1){
            if(j == 0){ h[i][j] = add(1, dif[u][j]); }
            else{ h[i][j] = add(dif[u][j], mul(j, dif[u][j - 1])); }
            continue;
         }
         for(l = 0, h[i][j] = 0; l <= min(sbtr[u], j); l++){
            x = mul(fac[j], mul(inv[l], inv[j - l]));
            if(l == 0){ x = mul(x, add(1, dif[u][0])); }
            else{ x = mul(x, add(dif[u][l], mul(l, dif[u][l - 1]))); }
            x = mul(x, h[i + 1][j - l]);
            h[i][j] = add(h[i][j], x);
         }
      }
   }
}
int F(int u, int v){
   int ret = 0;
   if(v > u){ return 0; }
   if(v == 0){ return 0; }
   if(u == 1){ return 1; }
```

```
58     if(f[u][v] != -1){ return f[u][v]; }
59     ret = add(mul(v, F(u - 1, v)), F(u - 1, v - 1));
60     return f[u][v] = ret;
61  }
62  bool cmp(int lhs, int rhs){ return sbtr[lhs] < sbtr[rhs]; }
63  void dfs(int src, int par){
64     int i, j, idx = -1, mns, ret, u, x; sbtr[src] = 1, ret = 0;
65     for(i = 0; i < adj[src].size(); i++){
66        u = adj[src][i];
67        if(u == par){ idx = i; break; }
68     }
69     if(idx != -1){ adj[src].erase(adj[src].begin() + idx); }
70     for(i = 0, dif[src][0] = 2, mns = 0; i < adj[src].size(); i++){
71        u = adj[src][i];
72        dfs(u, src);
73        sbtr[src] += sbtr[u];
74        dif[src][0] = mul(dif[src][0], add(1, dif[u][0]));
75     }
76     sort(adj[src].begin(), adj[src].end(), cmp);
77     dif[src][0] = add(dif[src][0], -1);
78     for(j = 0; j <= k; j++){
79        for(i = 0, sum[j] = 0; i < adj[src].size(); i++){
80           int u = adj[src][i];
81           sum[j] = add(sum[j], dif[u][j]);
82        }
83     }
84     H(src);
85     for(i = 1; i <= min(k, sbtr[src] - 1); i++){
86        u = adj[src][0];
87        x = mul(2, h[0][i]);
88        dif[src][i] = x;
89        x = add(x , -sum[i]);
90        x = add(x, -(mul(i, sum[i - 1])));//check korte hbe
91        x = mul(x, F(k, i));
92        ret = add(ret, x);
93     }
94     sol = add(sol, ret);
95  }
96  void solve(){
97     int i, j;
98     mem(f, -1); factorial();
99     dfs(0, -1);
100    pi(sol); nl;
101 }
102 int main(){
103 //   freopen("input.txt","r",stdin);
104 //   int ret = F(3, 3);
105 //   pi(ret); nl;
106 //   freopen("output.txt", "w", stdout);
107    input();
108    solve();
109 }
```