

```

1 // xor tree
2 // Problem: E. Swap and Maximum Block
3 // Contest: Codeforces - Educational Codeforces Round 133 (Rated for Div. 2)
4
5 #include "bits/stdc++.h"
6
7 struct node{
8     int pre, suf, sum, ans;
9 };
10
11 node merge(const node &a, const node &b){
12
13     node ret;
14     ret.sum = a.sum + b.sum;
15     ret.pre = max(a.pre, a.sum + b.pre);
16     ret.suf = max(b.suf, a.suf + b.sum);
17     ret.ans = max({a.ans, b.ans, a.suf+b.pre});
18
19     return ret;
20 }
21
22 int arr[NMAX];
23 vector<node> tree[NMAX << 2];
24
25 void build(int nd, int l, int r, int b){
26
27     if(l==r){
28         int tt = max(arr[l], 0ll);
29         tree[nd].push_back((node){tt, tt, arr[l], tt});
30         return;
31     }
32
33     int mid = (l+r)/2;
34     build(nd*2, l, mid, b-1);
35     build(nd*2+1, mid+1, r, b-1);
36
37     for(int x = 0; x < (1<<(b-1)); x++){
38         node a = tree[nd*2][x];
39         node b = tree[nd*2+1][x];
40         tree[nd].push_back(merge(a, b));
41     }
42
43     for(int x = 0; x < (1<<(b-1)); x++){
44         node a = tree[nd*2][x];
45         node b = tree[nd*2+1][x];
46         tree[nd].push_back(merge(b, a));
47     }
48 }
49
50 int32_t main(){
51
52     int n;
53     cin >> n;
54
55     for(int i = 0; i < (1<<n); i++){
56         cin >> arr[i];
57     }
58
59     build(1, 0, (1<<n)-1, n);
60
61     int x = 0;
62     int q; cin >> q; while(q--){
63
64         int cx;
65         cin >> cx;

```

```
66         x ^= 1<<cx;
67
68         cout << tree[1][x].ans << endl;
69     }
70 }
71
```

```

1 // xor segment tree v2 (queries with bs)
2 // Problem: F. Minimal String Xoration
3
4 // By AmmarDab3an
5
6 const int MAX = 2e5 + 10;
7 const int NMAX = (1<<18) + 10;
8 const int MMAX = 2e5 + 10;
9 const int LOG_MAX = ceil(log2(double(NMAX)));
10 const int BLOCK = ceil(sqrt(double(NMAX)));
11
12 int n;
13 string str;
14
15 array<int, 2> p, m;
16 vector<array<int, 2>> pow_m;
17
18 bool is_prime(int x){
19     for(ll i = 2; i*i <= x; i++) if(x%i==0){
20         return false;
21     }
22     return true;
23 }
24
25 void init_hash(){
26
27     p = {rand(1e4, 5e4), rand(6e4, 9e4)};
28     m = {53, 79};
29
30     while(!is_prime(p[0])) p[0]++;
31     while(!is_prime(p[1])) p[1]++;
32
33     pow_m.resize(NMAX);
34
35     pow_m[0][0] = pow_m[0][1] = 1;
36
37     for(int i = 1; i < NMAX; i++)
38         for(int j = 0; j < 2; j++){
39             pow_m[i][j] = (pow_m[i-1][j] * m[j])%p[j];
40         }
41 }
42
43 vector<array<int, 2>> tree[NMAX << 2];
44
45 void build(int nd, int l, int r, int p){
46
47     if(l==r){
48         array<int, 2> cur;
49         cur[0] = cur[1] = str[l]-'a'+1;
50         tree[nd].push_back(cur);
51         return;
52     }
53
54     int mid = (l+r)/2;
55     build(nd*2, l, mid, p-1);
56     build(nd*2+1, mid+1, r, p-1);
57
58     for(int i = 0; i < (1<<(p-1)); i++){
59
60         array<int, 2> a = tree[nd*2][i];
61         array<int, 2> b = tree[nd*2+1][i];
62
63         array<int, 2> cur;
64         for(int j = 0; j < 2; j++){
65             cur[j] = (a[j] * pow_m[1<<(p-1)][j] + b[j]) % ::p[j];

```

```

66     }
67     tree[nd].push_back(cur);
68 }
69
70 for(int i = 0; i < (1<<(p-1)); i++){
71
72     array<int, 2> a = tree[nd*2][i];
73     array<int, 2> b = tree[nd*2+1][i];
74
75     array<int, 2> cur;
76     for(int j = 0; j < 2; j++){
77         cur[j] = (b[j] * pow_m[1<<(p-1)][j] + a[j]) % ::p[j];
78     }
79     tree[nd].push_back(cur);
80 }
81 }
82
83 array<int, 2> query(int nd, int l, int r, int p, int x, int q_l, int q_r){
84
85
86     if(r < q_l || q_r < l){
87         return (array<int, 2>){0, 0};
88     }
89
90     if(q_l <= l && r <= q_r){
91
92         array<int, 2> cur = tree[nd][x];
93
94         for(int j = 0; j < 2; j++){
95             cur[j] = (cur[j] * pow_m[q_r-r][j]) % ::p[j];
96         }
97
98         return cur;
99     }
100
101     int mid = (l+r)/2;
102     array<int, 2> st_path, nd_path;
103
104     if((x>>(p-1))&1){
105         x ^= (1<<(p-1));
106         st_path = query(nd*2+1, l, mid, p-1, x, q_l, q_r);
107         nd_path = query(nd*2, mid+1, r, p-1, x, q_l, q_r);
108     }
109     else{
110         st_path = query(nd*2, l, mid, p-1, x, q_l, q_r);
111         nd_path = query(nd*2+1, mid+1, r, p-1, x, q_l, q_r);
112     }
113
114     array<int, 2> cur;
115     for(int j = 0; j < 2; j++){
116         cur[j] = (st_path[j] + nd_path[j]) % ::p[j];
117     }
118
119     return cur;
120 }
121
122 int query_bs(int nd_a, int nd_b, int l, int r, int a, int b, int p){
123
124     if(l==r){
125         if(tree[nd_a][0]!=tree[nd_b][0]){
126             return l;
127         }
128         else{
129             return l+1;
130         }

```

```

131     }
132
133     int mid = (l+r)/2;
134     int nd_lf_a = nd_a*2;
135     int nd_lf_b = nd_b*2;
136
137     if((a>>(p-1))&1) a ^= 1<<(p-1), nd_lf_a ^= 1;
138     if((b>>(p-1))&1) b ^= 1<<(p-1), nd_lf_b ^= 1;
139
140     if(tree[nd_lf_a][a] != tree[nd_lf_b][b]){
141         return query_bs(nd_lf_a, nd_lf_b, l, mid, a, b, p-1);
142     }
143     else{
144         return query_bs(nd_lf_a^1, nd_lf_b^1, mid+1, r, a, b, p-1);
145     }
146 }
147
148 bool comp(int a, int b){
149
150     int l = 0;
151     int r = (1<<n)-1;
152
153     // int ans = -1;
154
155     // while(l <= r){
156     //
157         // int mid = (l+r)/2;
158     //
159         // array<int, 2> hash_a = query(1, 0, (1<<n)-1, n, a, 0, mid);
160         // array<int, 2> hash_b = query(1, 0, (1<<n)-1, n, b, 0, mid);
161     //
162         // if(hash_a==hash_b){
163             // ans = mid;
164             // l = mid+1;
165         // }
166         // else{
167             // r = mid-1;
168         // }
169     //
170     // }
171
172     int ans = query_bs(1, 1, 0, (1<<n)-1, a, b, n) -1;
173
174     if(ans+1==(1<<n)){
175         return false;
176     }
177
178     char ch_a = str[(ans+1)^a];
179     char ch_b = str[(ans+1)^b];
180
181     return ch_a < ch_b;
182 }
183
184 int32_t main(){
185
186     cin >> n;
187     cin >> str;
188
189     init_hash();
190
191     build(1, 0, (1<<n)-1, n);
192
193     int ans = 0;
194     for(int i = 1; i < (1<<n); i++){
195         if(comp(i, ans)){

```

```
196         ans = i;
197     }
198 }
199
200 string ans_str(1<<n, '.');
201 for(int i = 0; i < (1<<n); i++){
202     ans_str[i] = str[i^ans];
203 }
204
205 cout << ans_str << endl;
206 }
207
```

```

1 // Problem: H. Codeforces Scoreboard
2 // Contest: Codeforces - TypeDB Forces 2023 (Div. 1 + Div. 2, Rated, Prizes!)
3
4 #include "bits/stdc++.h"
5
6 using namespace std;
7
8 struct node{
9
10     int size, dp, lazy;
11     node *lf, *ri;
12
13     ~node(){
14         delete lf;
15         delete ri;
16     }
17
18     bool is_leaf(){
19         assert((lf==0) == (size==1));
20         return size==1;
21     }
22
23     void push(){
24         if(!lazy) return;
25         dp += lazy;
26         if(lf) lf->lazy += lazy;
27         if(ri) ri->lazy += lazy;
28         lazy = 0;
29     }
30
31     void merge(){
32         assert(!is_leaf());
33         assert(lazy==0);
34         size = lf->size + ri->size;
35         dp = ri->dp;
36     }
37
38     bool is_balanced() {
39         assert(!is_leaf());
40         int l = lf->size;
41         int r = ri->size;
42         return (l <= 2*r+5) && (r <= 2*l+5);
43     }
44
45     void balance(){
46
47         assert(!is_leaf());
48         if(is_balanced()) return;
49
50         vi elements;
51         dfs(elements);
52         build(elements);
53     }
54
55     void build(vi elements){
56
57         lazy = 0;
58         size = elements.size();
59         dp = elements.back();
60
61         delete lf;
62         delete ri;
63
64         assert(!elements.empty());
65

```

```

66         if(elements.size()==1){
67             lf = ri = nullptr;
68             return;
69         }
70
71         lf = new node();
72         ri = new node();
73         auto mid = elements.begin() + elements.size()/2;
74         lf->build(vi(elements.begin(), mid));
75         ri->build(vi(mid, elements.end()));
76     }
77
78     bool wants(int i, int k, int b){
79         push();
80         int tt = b-(i+size-1)*k;
81         return tt > dp;
82     }
83
84     void update(int i, int k, int b){
85
86         push();
87
88         if(is_leaf()){
89             build({b-i*k, dp-k});
90             return;
91         }
92
93         if(lf->wants(i, k, b)){
94             lf->update(i, k, b);
95             ri->lazy -= k;
96             ri->push();
97         }
98         else{
99             ri->update(i+lf->size, k, b);
100         }
101
102         merge();
103         balance();
104     }
105
106     void dfs(vi &elements){
107
108         push();
109
110         if(is_leaf()){
111             elements.push_back(dp);
112         }
113         else{
114             lf->dfs(elements);
115             ri->dfs(elements);
116         }
117     }
118 };
119
120 int32_t main(){
121
122     fastIO;
123
124     int t; cin >> t; while(t--){
125
126         int n;
127         cin >> n;
128
129         int sm_a = 0;
130         vector<pii> vec(n);

```



```

131
132     for(auto &[k, b] : vec){
133         cin >> k >> b;
134         int ai;
135         cin >> ai;
136         sm_a += ai;
137         b -= ai;
138     }
139
140     sort(vec.rbegin(), vec.rend());
141
142     node rt = (node){1, -INFL, 0, nullptr, nullptr};
143
144     for(auto [k, b] : vec){
145         rt.update(1, k, b);
146     }
147
148     vi elements;
149     rt.dfs(elements);
150
151     int ans = sm_a;
152     for(auto e : elements){
153         if(e >= 0){
154             ans += e;
155         }
156     }
157
158     cout << ans << endl;
159 }
160 }
161

```

```

1 // sqrt cmp fst
2 // Problem: D. ! Divisible
3 // Contest: Codeforces - ACPC 2022
4
5 const int AMAX = 1e6 + 1e4;
6
7 const int NMAX = 2e5 + 10;
8 const int BLOCK = 650;
9 const int LOG_MAX = ceil(log2(double(NMAX)));
10
11 const int PMAX = 1e5 + 10;
12 const int PBLOCK = 300;
13 const int PMX = PMAX/PBLOCK + 1;
14
15 inline int64_t hilbertOrder(int x, int y, int pow, int rotate) {
16
17     if (pow == 0) {
18         return 0;
19     }
20     int hpow = 1 << (pow-1);
21     int seg = (x < hpow) ? (
22         (y < hpow) ? 0 : 3
23     ) : (
24         (y < hpow) ? 1 : 2
25     );
26     seg = (seg + rotate) & 3;
27     const int rotateDelta[4] = {3, 0, 0, 1};
28     int nx = x & (x ^ hpow), ny = y & (y ^ hpow);
29     int nrot = (rotate + rotateDelta[seg]) & 3;
30     int64_t subSquareSize = int64_t(1) << (2*pow - 2);
31     int64_t ans = seg * subSquareSize;
32     int64_t add = hilbertOrder(nx, ny, pow-1, nrot);
33     ans += (seg == 1 || seg == 2) ? add : (subSquareSize - add - 1);
34     return ans;
35 }
36
37 struct query{
38
39     int order;
40     int l, r, fix, idx;
41
42     bool operator < (const query &other){
43         return order < other.order;
44     }
45 };
46
47 vi adj[NMAX];
48 int val[NMAX];
49 vi primes;
50 bool not_prime[AMAX];
51 int prime_pos[AMAX];
52 int tin[NMAX], tout[NMAX], tim;
53 int arr[2*NMAX];
54
55 bool vis[NMAX];
56 int pcnt[PMAX];
57 int blk_sz[PMX];
58 int blk_osz[PMX];
59
60 int depth[NMAX];
61 int par[NMAX][LOG_MAX];
62
63 query queries[NMAX];
64 int ans[NMAX];
65

```

```

66 void init_sieve(){
67
68     for(ll i = 2; i < AMAX; i++) if(!not_prime[i]){
69         prime_pos[i] = primes.size();
70         primes.push_back(i);
71         for(ll j = i*i; j < AMAX; j+=i){
72             not_prime[j] = true;
73         }
74     }
75
76     for(int i = 0; i < primes.size(); i++){
77         blk_osz[i/PBLOCK]++;
78     }
79 }
80
81 void dfs(int u, int p){
82
83     tin[u] = tim;
84     arr[tim] = u;
85     tim++;
86
87     for(auto v : adj[u]) if(v != p){
88
89         depth[v] = depth[u]+1;
90
91         par[v][0] = u;
92         for(int i = 1; i < LOG_MAX; i++){
93             par[v][i] = par[par[v][i-1]][i-1];
94         }
95
96         dfs(v, u);
97     }
98
99     tout[u] = tim;
100    arr[tim] = u;
101    tim++;
102 }
103
104 int lca(int u, int v){
105
106     if(depth[u] < depth[v]){
107         swap(u, v);
108     }
109
110     int dif = depth[u] - depth[v];
111
112     for(int i = 0; i < LOG_MAX; i++) if((dif>>i)&1){
113         u = par[u][i];
114     }
115
116     if(u==v) return u;
117
118     for(int i = LOG_MAX-1; i >= 0; i--) if(par[u][i] != par[v][i]){
119         u = par[u][i];
120         v = par[v][i];
121     }
122
123     return par[u][0];
124 }
125
126 void update(int f, int d){
127
128     pcnt[f] += d;
129
130     if(pcnt[f]==1){

```

```

131         blk_sz[f/PBLOCK] += 1;
132     }
133     else if(pcnt[f]==0){
134         blk_sz[f/PBLOCK] -= 1;
135     }
136 }
137
138 void add(int i){
139
140     int u = arr[i];
141     if(val[u]==-1) return;
142
143     vis[u] ^= 1;
144     update(val[u], vis[u] ? +1 : -1);
145 }
146
147 void rem(int i){
148     add(i);
149 }
150
151 int get_ans(){
152
153     for(int i = 0; i < PMX; i++) if(blk_sz[i] != blk_osz[i]){
154
155         assert(blk_sz[i] < blk_osz[i]);
156
157         int lo = i*PBLOCK;
158         int hi = (i+1)*PBLOCK;
159
160         for(int j = lo; j < hi; j++){
161
162             assert(j < PMAX);
163
164             if(pcnt[j]==0){
165                 return primes[j];
166             }
167         }
168     }
169
170     assert(false);
171 }
172
173 int32_t main(){
174
175     fastIO;
176
177     init_sieve();
178
179     int t; cin >> t; while(t--){
180
181         int n;
182         cin >> n;
183
184         {
185             tim = 0;
186             for(int i = 0; i < n; i++){
187                 adj[i].clear();
188             }
189         }
190
191         for(int i = 0; i < n; i++){
192             int x;
193             cin >> x;
194             val[i] = !not_prime[x] ? prime_pos[x] : -1;
195         }

```

```

196
197     for(int i = 1; i < n; i++){
198
199         int u, v;
200         cin >> u >> v;
201         u--, v--;
202
203         adj[u].push_back(v);
204         adj[v].push_back(u);
205     }
206
207     dfs(0, -1);
208
209     int q;
210     cin >> q;
211
212     for(int i = 0; i < q; i++){
213
214         int u, v;
215         cin >> u >> v;
216         u--, v--;
217
218         if(depth[u] > depth[v]){
219             swap(u, v);
220         }
221
222         int p = lca(u, v);
223
224         if(u==p){
225             queries[i] = {0, tin[u], tin[v], -1, i};
226         }
227         else{
228             queries[i] = {0, tout[u], tin[v], tin[p], i};
229         }
230     }
231
232     for(int i = 0; i < q; i++){
233         int l = queries[i].l;
234         int r = queries[i].r;
235         queries[i].order = hilbertOrder(l, r, 20, 0);
236     }
237
238     sort(queries, queries+q);
239
240     int l = 1, r = 0;
241
242     for(int i = 0; i < q; i++){
243
244         auto [co, cl, cr, fix, idx] = queries[i];
245
246         while(cl < l) add(--l);
247         while(r < cr) add(++r);
248         while(l < cl) rem(l++);
249         while(cr < r) rem(r--);
250
251         if(fix != -1){
252             add(fix);
253         }
254
255         ans[idx] = get_ans();
256
257         if(fix != -1){
258             rem(fix);
259         }
260     }

```

```
261
262     while(r >= 1){
263         rem(r--);
264     }
265
266     for(int i = 0; i < q; i++){
267         cout << ans[i] << endl;
268     }
269 }
270 }
271
```

```

1 // crt
2 // Problem: C. Counting Trees
3 // Contest: Codeforces - NUS CS3233 Final Team Contest 2023 Mirror
4
5 #include "bits/stdc++.h"
6
7 using namespace std;
8
9 const int MOD = 3000301; // 1e9 + 7;
10 const int NMAX = MOD;
11
12 int fac[NMAX], ifac[NMAX];
13 int mod = 10000003233;
14 vi mod_factors = {3, 11, 101, 3000301};
15 int mem[3][222][222];
16
17 void init(){
18
19     fac[0] = 1;
20     for(int i = 1; i < NMAX; i++){
21         fac[i] = mul(fac[i-1], i);
22     }
23
24     ifac[NMAX-1] = inv(fac[NMAX-1]);
25     for(int i = NMAX-2; i >= 0; i--){
26         ifac[i] = mul(ifac[i+1], i+1);
27     }
28
29     memset(mem, -1, sizeof mem);
30 }
31
32 int cho(int n, int c, int i){
33
34     if(n == c) return 1;
35     if(n == 0) return 0;
36
37     int &ret = mem[i][n][c];
38     if(ret != -1) return ret;
39
40     int st_path = cho(n-1, c-1, i);
41     int nd_path = cho(n-1, c, i);
42     int ans = (st_path + nd_path) % mod_factors[i];
43
44     return ret = ans;
45 }
46
47 int choose(int n, int c, int i){
48
49     if(n < c) return 0;
50
51     if(i < 3){
52         return cho(n, c, i);
53     }
54     else{
55         return mul(fac[n], mul(ifac[c], ifac[n-c]));
56     }
57 }
58
59 struct Congruence {
60     long long a, m;
61 };
62
63 long long chinese_remainder_theorem(vector<Congruence> const& congruences) {
64
65     long long M = 1;

```

```

66     for (auto const& congruence : congruences) {
67         M *= congruence.m;
68     }
69
70     long long solution = 0;
71     for (auto const& congruence : congruences) {
72         long long a_i = congruence.a;
73         long long M_i = M / congruence.m;
74         long long N_i = pow_exp(M_i, congruence.m-2, congruence.m);
75         solution = (solution + a_i * M_i % M * N_i) % M;
76     }
77
78     return solution;
79 }
80
81 int32_t main() {
82
83     init();
84
85     int t; cin >> t; while(t--){
86
87         int n, k, c;
88         cin >> n >> k >> c;
89
90         if(c > n){
91             cout << 0 << endl;
92             continue;
93         }
94
95         vector<Congruence> vec(4);
96
97         for(int i = 0; i < 4; i++){
98
99             int cm = mod_factors[i];
100
101             int cho = 1;
102             int a = n-1;
103             int b = c-1;
104
105             while(a > 0){
106                 int ccho = choose(a%cm, b%cm, i);
107                 cho = (cho * ccho)%cm;
108                 a /= cm, b /= cm;
109             }
110
111             vec[i] = {cho, cm};
112         }
113
114         int ans = chinese_remainder_theorem(vec);
115         ans = (ans * 2) % mod;
116
117         cout << ans << endl;
118     }
119 }
120

```



```

1 // x_factor int
2 // I. Investors
3 // https://qoj.ac/contest/1103/problem/5507
4
5 // By AmmarDab3an
6
7 const int MAX = 2e5 + 10;
8 const int NMAX = 2e5 + 10;
9 const int MMAX = 2e5 + 10;
10 const int LOG_MAX = ceil(log2(double(NMAX)));
11 const int BLOCK = ceil(sqrt(double(NMAX)));
12
13 struct FenwickTree {
14     vector<int> bit; // binary indexed tree
15     int n;
16
17     FenwickTree(int n) {
18         this->n = n;
19         bit.assign(n, 0);
20     }
21
22     FenwickTree(vector<int> a) : FenwickTree(a.size()) {
23         for (size_t i = 0; i < a.size(); i++)
24             add(i, a[i]);
25     }
26
27     int sum(int r) {
28         int ret = 0;
29         for (; r >= 0; r = (r & (r + 1)) - 1)
30             ret += bit[r];
31         return ret;
32     }
33
34     int sum(int l, int r) {
35         return sum(r) - sum(l - 1);
36     }
37
38     void add(int idx, int delta) {
39         for (; idx < n; idx = idx | (idx + 1))
40             bit[idx] += delta;
41     }
42 };
43
44 int pre[6060][6060];
45
46 int32_t main() {
47     fastIO;
48
49     int t; cin >> t; while(t--){
50
51         int n, k;
52         cin >> n >> k;
53
54         k++;
55
56         vi vec(n);
57         for(auto &i : vec) cin >> i;
58
59         if(n==0){
60             srand(0);
61             n = 6000;
62             vec = vi(n);
63             iota(vec.begin(), vec.end(), 0);
64             random_shuffle(vec.begin(), vec.end());
65

```

```

66         k = 5 + 1;
67     }
68
69     vi tmp = vec;
70     sort(tmp.begin(), tmp.end());
71     tmp.erase(unique(tmp.begin(), tmp.end()), tmp.end());
72
73     for(auto &e : vec){
74         e = lower_bound(tmp.begin(), tmp.end(), e) - tmp.begin();
75     }
76
77     for(int i = 0; i < n; i++){
78         FenwickTree bit(n);
79         int cnt = 0;
80         for(int j = i; j >= 0; j--){
81             cnt += bit.sum(0, vec[j]-1);
82             bit.add(vec[j], 1);
83             pre[i][j] = cnt;
84         }
85     }
86
87     auto calc = [&](int x_factor)->pii{
88
89         vpii dp(n);
90
91         for(int i = 0; i < n; i++){
92
93             pii cans = {INFL, 0};
94
95             for(int j = i; j >= 0; j--){
96
97                 pii nxt = j ? dp[j-1] : (pii){0, 0};
98                 nxt.first += pre[i][j] + x_factor;
99                 nxt.second++;
100
101                 cans = min(cans, nxt);
102             }
103
104             dp[i] = cans;
105         }
106
107         return dp[n-1];
108     };
109
110     int l = 0;
111     int r = INF;
112
113     int bs_ans = -1;
114
115     while(l <= r){
116
117         int mid = (l+r)/2;
118
119         pii cans = calc(mid);
120
121         if(cans.second <= k){
122             bs_ans = mid;
123             r = mid-1;
124         }
125         else{
126             l = mid+1;
127         }
128     }
129
130     pii ans = calc(bs_ans);

```

```
131         cout << ans.first - bs_ans*k << endl;
132     }
133 }
134
```

```

1 // x_factor double
2 // Problem: E. Gosha is hunting
3
4 #include "bits/stdc++.h"
5
6 pair<double, int> merge(const pair<double, int> &a, const pair<double, int> &b){
7     if(abs(a.first-b.first) < 1e-6){
8         return a.second < b.second ? a : b;
9     }
10    else{
11        return a.first > b.first ? a : b;
12    }
13 }
14
15 int32_t main(){
16
17     int n, a, b;
18     cin >> n >> a >> b;
19
20     vector<vector<double>> vec(2, vector<double>(n));
21     for(auto &v : vec) for(auto &i : v) cin >> i;
22
23     auto calc = [&](double x_factor){
24
25         vector<pair<double, int>> dp(a+1);
26
27         for(int i = 0; i < n; i++){
28
29             vector<pair<double, int>> ndp(a+1, {-1e18, INF});
30
31             double p = vec[0][i];
32             double q = vec[1][i];
33             double pq = 1.0 - (1.0-p)*(1.0-q);
34
35             for(int j = 0; j <= a; j++){
36
37                 // dp[i][a] = max(
38                     // dp[i-1][a],
39                     // dp[i-1][a-1] + vec[0][i]
40                     // dp[i-1][a] + vec[1][i] - x_factor
41                     // dp[i-1][a-1] + (1 - (1-vec[0][i])*(1-vec[1][i])) - x_factor
42                 // )
43
44                 auto &cans = ndp[j];
45
46                 auto st_path = dp[j];
47                 auto nd_path = dp[j];
48                 nd_path.first += q - x_factor;
49                 nd_path.second += 1;
50
51                 cans = merge(cans, merge(st_path, nd_path));
52
53                 if(j){
54                     auto rd_path = dp[j-1];
55                     rd_path.first += p;
56                     auto th_path = dp[j-1];
57                     th_path.first += pq - x_factor;
58                     th_path.second += 1;
59                     cans = merge(cans, merge(rd_path, th_path));
60                 }
61             }
62
63             dp = ndp;
64         }
65

```

```

66         return dp.back();
67     };
68
69     double l = 0;
70     double r = 1e9;
71
72     double bs_ans = -1;
73
74     int cnt = 100;
75     while(cnt--){
76
77         double mid = (l+r)/2;
78         auto cans = calc(mid);
79
80         if(cans.second <= b){
81             bs_ans = mid;
82             r = mid;
83         }
84         else{
85             l = mid;
86         }
87     }
88
89     double ans = calc(bs_ans).first + b*bs_ans;
90
91     cout << fixed << setprecision(6) << ans << endl;
92 }
93

```

```

1 // dp_schoelace
2 // B. Bars
3 // https://qoj.ac/problem/5500
4
5 // By AmmarDab3an
6
7 #include "bits/stdc++.h"
8
9 int32_t main(){
10
11     int t; cin >> t; while(t--){
12
13         int n;
14         cin >> n;
15
16         vi vec(n);
17         for(auto &i : vec) cin >> i;
18
19         pii tmp;
20         tmp.push_back({0, 0});
21
22         // dp[i] = max(dp[j] + (pi+pj)*(i-j))
23         // dp[i] = max(dp[j] + pi*i + pj*j - pi*j - pj*j)
24         // dp[i] = pi*i + max((dp[j]-pj*j) + pj*i - pi*j )
25
26         // ans = sum((pi+pj) * (i-j))
27         // Shoelace Formula
28
29         auto calc = [&](const pii &a, const pii &b){
30             return a.first*b.second - b.first*a.second;
31         };
32
33         for(int i = 0; i <= n; i++){
34
35             pii cur = i < n ? (pii){vec[i], i} : (pii){0, n-1};
36
37             while(tmp.size() >= 2){
38
39                 pii a = tmp[tmp.size()-2];
40                 pii b = tmp[tmp.size()-1];
41                 pii c = cur;
42
43                 if(calc(a, b) + calc(b, c) <= calc(a, c)){
44                     tmp.pop_back();
45                 }
46                 else{
47                     break;
48                 }
49             }
50
51             tmp.push_back(cur);
52         }
53
54         int ans = 0;
55         for(int i = 1; i < tmp.size(); i++){
56             ans += calc(tmp[i-1], tmp[i]);
57         }
58
59         cout << ans << endl;
60     }
61 }
62

```

```

1  // fft any mod
2
3  #include<bits/stdc++.h>
4  using namespace std;
5  #define LL long long
6  using namespace std;
7  using cd = complex < long double >;
8  long double PI = acos ( - 1 );
9  long long mod = 1e9 + 7;
10 namespace fft{
11     struct num{
12         double x,y;
13         num() {x=y=0;}
14         num(double x,double y):x(x),y(y){}
15     };
16     inline num operator+(num a,num b) {return num(a.x+b.x,a.y+b.y);}
17     inline num operator-(num a,num b) {return num(a.x-b.x,a.y-b.y);}
18     inline num operator*(num a,num b) {return num(a.x*b.x-a.y*b.y,a.x*b.y+a.y*b.x);}
19     inline num conj(num a) {return num(a.x,-a.y);}
20     int base=1;
21     vector<num> roots={{0,0},{1,0}};
22     vector<int> rev={0,1};
23     const double PI=acosl(-1.0);
24     void ensure_base(int nbase){
25         if(nbase<=base) return;
26         rev.resize(1<<nbase);
27         for(int i=0;i<(1<<nbase);i++)
28             rev[i]=(rev[i>>1]>>1)+((i&1)<<(nbase-1));
29         roots.resize(1<<nbase);
30         while(base<nbase){
31             double angle=2*PI/(1<<(base+1));
32             for(int i=1<<(base-1);i<(1<<base);i++){
33                 roots[i<<1]=roots[i];
34                 double angle_i=angle*(2*i+1-(1<<base));
35                 roots[(i<<1)+1]=num(cos(angle_i),sin(angle_i));
36             }
37             base++;
38         }
39     }
40
41     void fft(vector<num> &a,int n=-1){
42         if(n==-1) n=a.size();
43         assert((n&(n-1))==0);
44         int zeros=__builtin_ctz(n);
45         ensure_base(zeros);
46         int shift=base-zeros;
47         for(int i=0;i<n;i++)
48             if(i<(rev[i]>>shift))
49                 swap(a[i],a[rev[i]>>shift]);
50         for(int k=1;k<n;k<=<1){
51             for(int i=0;i<n;i+=2*k){
52                 for(int j=0;j<k;j++){
53                     num z=a[i+j+k]*roots[j+k];
54                     a[i+j+k]=a[i+j]-z;
55                     a[i+j]=a[i+j]+z;
56                 }
57             }
58         }
59     }
60     vector<num> fa,fb;
61     vector<int> multiply(vector<int> &a, vector<int> &b){
62         int need=a.size()+b.size()-1;
63         int nbase=0;
64         while((1<<nbase)<need) nbase++;
65         ensure_base(nbase);

```

```

66     int sz=1<<nbase;
67     if(sz>(int)fa.size()) fa.resize(sz);
68     for(int i=0;i<sz;i++){
69         int x=(i<(int)a.size()?a[i]:0);
70         int y=(i<(int)b.size()?b[i]:0);
71         fa[i]=num(x,y);
72     }
73     fft(fa,sz);
74     num r(0,-0.25/sz);
75     for(int i=0;i<=(sz>>1);i++){
76         int j=(sz-i)&(sz-1);
77         num z=(fa[j]*fa[j]-conj(fa[i]*fa[i]))*r;
78         if(i!=j) fa[j]=(fa[i]*fa[i]-conj(fa[j]*fa[j]))*r;
79         fa[i]=z;
80     }
81     fft(fa,sz);
82     vector<int> res(need);
83     for(int i=0;i<need;i++) res[i]=fa[i].x+0.5;
84     return res;
85 }
86
87 vector<int> multiply_mod(vector<int> &a,vector<int> &b,int m,int eq=0){
88     int need=a.size()+b.size()-1;
89     int nbase=0;
90     while((1<<nbase)<need) nbase++;
91     ensure_base(nbase);
92     int sz=1<<nbase;
93     if(sz>(int)fa.size()) fa.resize(sz);
94     for(int i=0;i<(int)a.size();i++){
95         int x=(a[i]%m+m)%m;
96         fa[i]=num(x&((1<<15)-1),x>>15);
97     }
98     fill(fa.begin()+a.size(),fa.begin()+sz,num{0,0});
99     fft(fa,sz);
100    if(sz>(int)fb.size()) fb.resize(sz);
101    if(eq) copy(fa.begin(),fa.begin()+sz,fb.begin());
102    else{
103        for(int i=0;i<(int)b.size();i++){
104            int x=(b[i]%m+m)%m;
105            fb[i]=num(x&((1<<15)-1),x>>15);
106        }
107        fill(fb.begin()+b.size(),fb.begin()+sz,num{0,0});
108        fft(fb,sz);
109    }
110    double ratio=0.25/sz;
111    num r2(0,-1),r3(ratio,0),r4(0,-ratio),r5(0,1);
112    for(int i=0;i<=(sz>>1);i++){
113        int j=(sz-i)&(sz-1);
114        num a1=(fa[i]+conj(fa[j]));
115        num a2=(fa[i]-conj(fa[j]))*r2;
116        num b1=(fb[i]+conj(fb[j]))*r3;
117        num b2=(fb[i]-conj(fb[j]))*r4;
118        if(i!=j){
119            num c1=(fa[j]+conj(fa[i]));
120            num c2=(fa[j]-conj(fa[i]))*r2;
121            num d1=(fb[j]+conj(fb[i]))*r3;
122            num d2=(fb[j]-conj(fb[i]))*r4;
123            fa[i]=c1*d1+c2*d2*r5;
124            fb[i]=c1*d2+c2*d1;
125        }
126        fa[j]=a1*b1+a2*b2*r5;
127        fb[j]=a1*b2+a2*b1;
128    }
129    fft(fa,sz);fft(fb,sz);
130    vector<int> res(need);

```



```

131         for(int i=0;i<need;i++){
132             LL aa=fa[i].x+0.5;
133             LL bb=fb[i].x+0.5;
134             LL cc=fa[i].y+0.5;
135             res[i]=(aa+((bb%m)<<15)+((cc%m)<<30))%m;
136         }
137         return res;
138     }
139     vector<int> square_mod(vector<int> &a,int m){
140         return multiply_mod(a,a,m,1);
141     }
142 };
143

```

```

1 // mob_baath
2 // Problem: I. Will you accept Basharo challenge?
3 // Contest: Codeforces - Al-Baath Collegiate Programming Contest 2023
4
5 #include "bits/stdc++.h"
6
7 const int NMAX = 5e4 + 10;
8 const int AMAX = 5e4 + 10;
9
10 int arr[NMAX];
11 vi adj[NMAX];
12 int ans[NMAX], cans;
13 int frq0[AMAX];
14 int frq1[AMAX];
15 vi factors[AMAX];
16 int sub[NMAX];
17 map<pii, int> edge_id;
18
19 vector<int> prime;
20 bool not_prime[AMAX];
21 int mob[AMAX];
22
23 void mobius(int n = AMAX){
24
25     mob[1] = 1;
26
27     for(int i = 2; i < n; i++){
28
29         if(!not_prime[i]){
30             prime.push_back(i);
31             mob[i] = -1;
32         }
33
34         for (int j = 0; j < prime.size () && i * prime[j] < n; ++j) {
35
36             not_prime[i * prime[j]] = true;
37
38             if (i % prime[j] == 0){
39                 mob[i * prime[j]] = 0;
40                 break;
41             }
42             else{
43                 mob[i * prime[j]] = mob[i] * mob[prime[j]];
44             }
45         }
46     }
47 }
48
49 void init_factors(){
50     for(int i = 1; i < AMAX; i++)
51         for(int j = i; j < AMAX; j+=i){
52             factors[j].push_back(i);
53         }
54 }
55
56 void dfs(int u, int p){
57     sub[u] = 1;
58     if(p != -1) adj[u].erase(find(adj[u].begin(), adj[u].end(), p));
59     for(auto &v : adj[u]) if(v != p){
60         dfs(v, u);
61         sub[u] += sub[v];
62         if(sub[v] > sub[adj[u][0]]){
63             swap(v, adj[u][0]);
64         }
65     }

```

```

66     }
67
68     void add(int u, int d){
69         for(auto f : factors[arr[u]]){
70             cans -= mob[f] * (frq0[f]-frq1[f]) * frq1[f];
71             frq1[f] += d;
72             cans += mob[f] * (frq0[f]-frq1[f]) * frq1[f];
73         }
74     }
75
76     void add(int u, int p, int d){
77         add(u, d);
78         for(auto v : adj[u]) if(v != p){
79             add(v, u, d);
80         }
81     }
82
83     void calc(int u, int p, bool keep){
84
85         for(auto v : adj[u]) if(v != p) if(v != adj[u][0]){
86             calc(v, u, 0);
87         }
88
89         if(!adj[u].empty()){
90             calc(adj[u][0], u, 1);
91         }
92
93         for(auto v : adj[u]) if(v != p) if(v != adj[u][0]){
94             add(v, u, 1);
95         }
96
97         add(u, 1);
98         if(p != -1) ans[edge_id[{u, p}]] = cans;
99
100        if(!keep){
101            add(u, p, -1);
102        }
103    }
104
105    int32_t main(){
106
107        fastIO;
108
109        int n;
110        cin >> n;
111
112        mobius();
113        init_factors();
114
115        for(int i = 0; i < n; i++){
116            cin >> arr[i];
117            for(auto f : factors[arr[i]]){
118                frq0[f]++;
119            }
120        }
121
122        for(int i = 1; i < n; i++){
123            int u, v;
124            cin >> u >> v;
125            u--, v--;
126            adj[u].push_back(v);
127            adj[v].push_back(u);
128            edge_id[{u, v}] = edge_id[{v, u}] = i;
129        }
130

```

```

131     if(n==0){
132         n = 5e4;
133         // fill(arr, arr+n, 1);
134         iota(arr, arr+n, 1);
135         for(int i = 0; i < n; i++){
136             for(auto f : factors[arr[i]]){
137                 frq0[f]++;
138             }
139         }
140         rng = mt19937(0);
141         for(int i = 1; i < n; i++){
142             adj[i-1].push_back(i);
143             adj[i].push_back(i-1);
144             edge_id[{i-1, i}] = edge_id[{i, i-1}] = i;
145         }
146     }
147
148     dfs(0, -1);
149     calc(0, -1, 0);
150
151     for(int i = 1; i < n; i++){
152         cout << ans[i] << ' ';
153     }
154 }
155

```