```cpp
#include <bits/stdc++.h>
using namespace std;

const int MAXN = 40005;
const int MAXM = 100005;
const int LN = 19;

int N, M, K, cur, A[MAXN], LVL[MAXN], DP[LN][MAXN];
int BL[MAXN << 1], ID[MAXN << 1], VAL[MAXN], ANS[MAXM];
int d[MAXN], l[MAXN], r[MAXN];
bool VIS[MAXN];
vector < int > adjList[MAXN];

struct query{
    int id, l, r, lc;
    bool operator < (const query& rhs){
        return (BL[l] == BL[rhs.l]) ? (r < rhs.r) : (BL[l] < BL[rhs.l]);
    }
}Q[MAXM];

// Set up Stuff
void dfs(int u, int par){
    l[u] = ++cur;
    ID[cur] = u;
    for (int i = 1; i < LN; i++) DP[i][u] = DP[i - 1][DP[i - 1][u]];
    for (int i = 0; i < adjList[u].size(); i++){
        int v = adjList[u][i];
        if (v == par) continue;
        LVL[v] = LVL[u] + 1;
        DP[0][v] = u;
        dfs(v, u);
    }
    r[u] = ++cur; ID[cur] = u;
}

// Function returns lca of (u) and (v)
inline int lca(int u, int v){
    if (LVL[u] > LVL[v]) swap(u, v);
    for (int i = LN - 1; i >= 0; i--)
        if (LVL[v] - (1 << i) >= LVL[u]) v = DP[i][v];
    if (u == v) return u;
    for (int i = LN - 1; i >= 0; i--){
        if (DP[i][u] != DP[i][v]){
            u = DP[i][u];
            v = DP[i][v];
        }
    }
    return DP[0][u];
}

inline void check(int x, int& res){
    // If (x) occurs twice, then don't consider it's value
    if ( (VIS[x]) and (--VAL[A[x]] == 0) ) res--;
    else if ( (!VIS[x]) and (VAL[A[x]]++ == 0) ) res++;
    VIS[x] ^= 1;
}

void compute(){

    // Perform standard Mo's Algorithm
    int curL = Q[0].l, curR = Q[0].l - 1, res = 0;

    for (int i = 0; i < M; i++){

        while (curL < Q[i].l) check(ID[curL++], res);
        while (curL > Q[i].l) check(ID[--curL], res);
        while (curR < Q[i].r) check(ID[++curR], res);
        while (curR > Q[i].r) check(ID[curR--], res);

        int u = ID[curL], v = ID[curR];
```

```
71
72          // Case 2
73          if (Q[i].lc != u and Q[i].lc != v) check(Q[i].lc, res);
74
75          ANS[Q[i].id] = res;
76
77          if (Q[i].lc != u and Q[i].lc != v) check(Q[i].lc, res);
78      }
79
80      for (int i = 0; i < M; i++) printf("%d\n", ANS[i]);
81  }
82
83  int main(){
84
85      int u, v, x;
86
87      while (scanf("%d %d", &N, &M) != EOF){
88
89          // Cleanup
90          cur = 0;
91          memset(VIS, 0, sizeof(VIS));
92          memset(VAL, 0, sizeof(VAL));
93          for (int i = 1; i <= N; i++) adjList[i].clear();
94
95          // Inputting Values
96          for (int i = 1; i <= N; i++) scanf("%d", &A[i]);
97          memcpy(d + 1, A + 1, sizeof(int) * N);
98
99          // Compressing Coordinates
100         sort(d + 1, d + N + 1);
101         K = unique(d + 1, d + N + 1) - d - 1;
102         for (int i = 1; i <= N; i++) A[i] = lower bound(d + 1, d + K + 1, A[i]) - d;
103
104         // Inputting Tree
105         for (int i = 1; i < N; i++){
106             scanf("%d %d", &u, &v);
107             adjList[u].push back(v);
108             adjList[v].push back(u);
109         }
110
111         // Preprocess
112         DP[0][1] = 1;
113         dfs(1, -1);
114         int size = sqrt(cur);
115
116         for (int i = 1; i <= cur; i++) BL[i] = (i - 1) / size + 1;
117
118         for (int i = 0; i < M; i++){
119             scanf("%d %d", &u, &v);
120             Q[i].lc = lca(u, v);
121             if (l[u] > l[v]) swap(u, v);
122             if (Q[i].lc == u) Q[i].l = l[u], Q[i].r = l[v];
123             else Q[i].l = r[u], Q[i].r = l[v];
124             Q[i].id = i;
125         }
126
127         sort(Q, Q + M);
128         compute();
129     }
130 }
131
```