//#define \_CRT\_SECURE\_NO\_WARNINGS 1

#include <iostream>

#include <cstring>

#include <string>

#include <algorithm>

#include <queue>

#include <stack>

#include <vector>

#include <map>

#include <set>

#include <list>

#include <cstdio>

#include <utility>

#include <bitset>

#include <ctime>

#include <random>

#include <iomanip>

#include <cmath>

using namespace std;

#define IOS ios::sync\_with\_stdio(false);cin.tie(nullptr);cout.tie(nullptr);

#define endl '\n'

#define mem(f,z) memset(f,z,sizeof f)

#define INF 0x7fffffff

#define INFF 0x3f3f3f3f

#define ll long long

#define ull unsigned long long

#define PP pair<int ,int>

struct UF

{

vector<int> par;

UF(int n) :par(n)

{

for (int i = 0; i < n; i++)

par[i] = i;

}

int find(int x)

{

return x == par[x] ? x : par[x] = find(par[x]);

}

};

struct edge

{

int a, b, c;

bool operator<(const edge& other)const

{

return c < other.c;

}

};

int main()

{

IOS;

int N, M, Q;

cin >> N >> M >> Q;

vector<int> A(N + N);

for (int i = 0; i < N; i++)

{

cin >> A[i];

}

vector<edge> edges(M);

for (auto& e : edges)

{

cin >> e.a >> e.b >> e.c;

e.a--;

e.b--;

}

sort(edges.begin(), edges.end());

const int NN = N + N;

const int LOG = 20;

vector<vector<int>> parent(LOG, vector<int>(NN, NN - 2));

vector<vector<int>> extra(LOG, vector<int>(NN, INF));

vector<int> subtree(NN, 0);

UF uf(NN);

int cur = N;

for (int i = 0; i < N; i++)

subtree[i] = A[i];

for (auto& e : edges)

{

int a = uf.find(e.a), b = uf.find(e.b);

if (a == b)

continue;

uf.par[a] = uf.par[b] = uf.par[cur] = cur;

subtree[cur] = subtree[a] + subtree[b];

parent[0][a] = cur;

parent[0][b] = cur;

extra[0][a] = e.c - subtree[a];

extra[0][b] = e.c - subtree[b];

A[cur++] = e.c;

}

for (int j = 1; j < LOG; j++)

{

for (int i = 0; i < N + N; i++)

{

parent[j][i] = parent[j - 1][parent[j - 1][i]];

extra[j][i] = max(extra[j - 1][i], extra[j - 1][parent[j - 1][i]]);

}

}

for (int q = 0; q < Q; q++)

{

int node;

ll k;

cin >> node >> k;

node--;

for (int i = LOG - 1; i >= 0; i--)

{

if (k >= extra[i][node])

node = parent[i][node];

}

cout << k + subtree[node] << endl;

}

return 0;

}