```cpp
#include<stdio.h>
#include<string.h>
#include<algorithm>
#include<utility>
#include<vector>
#include<queue>
using namespace std;

#define MAX 112345
#define MAXS 512345
#define INF 1123456789
#define DEBP

typedef struct aresta{ int u, v, c; aresta() {}
  aresta(int _u, int _v, int _c) : u(_u), v(_v), c(_c) {} }aresta;
typedef pair<int, int> ii;
typedef vector<ii> vii;
typedef vector<int> vi;

int n, m, seen[MAXS], resp[MAXS], cont = 0;
int pai[MAX], tam[MAX];
aresta E[MAX];
ii query[MAXS];

void Make(int x) { pai[x] = x; tam[x] = 0; }
int Find(int x) { return pai[x] == x ? x : pai[x] = Find(pai[x]); }
void Union(int x, int y, int valor, vector<vi> &indices) {
  int i, a;
  if (tam[y] > tam[x]) return Union(y, x, valor, indices);
  if (tam[x] == tam[y]) tam[x]++;
  pai[y] = pai[x];
  for (i = 0; i < (int)indices[y].size(); i++) {
    a = indices[y][i];
    if ((query[a].first == x || query[a].second == x) && !seen[a]) {
      seen[a] = 1; resp[a] = valor;
    } else if (!seen[a]) {
      indices[x].push_back(a);
      if (query[a].first == y) query[a].first = x;
      else query[a].second = x;
    }}}

int cmp(const void *a, const void *b) {
  aresta *e = (aresta *)a, *f = (aresta *)b;
  return f->c - e->c;
}

void kruskal(vector<vi> &indices) {
  int u, v, i;
  for (i = 0; i < m; i++) {
    u = E[i].u; v = E[i].v;
    if (Find(u) != Find(v)) Union(Find(u), Find(v), E[i].c, indices);
  }}

int main(void) {
  int s, i, a, b, p;
  while (scanf("%d %d %d", &n, &m, &s) != EOF) {
    for (i = 0; i < n; i++) Make(i);
    for (i = 0; i < m; i++) {
      scanf("%d %d %d", &a, &b, &p); a--; b--;
      E[i] = aresta(a, b, p);
    }
    qsort(E, m, sizeof(aresta), &cmp);
    vector<vi> indices(n);
    memset(seen, 0, sizeof(seen));
    for (i = 0; i < s; i++) {
      scanf("%d %d", &a, &b); a--; b--;
      query[i] = ii(a, b);
      indices[a].push_back(i); indices[b].push_back(i);
    }
    kruskal(indices);
    for (i = 0; i < s; i++) printf("%d\n", resp[i]);
  }
  return 0;
}
```