matita.cpp

```
// Alessandro Guerrieri, fork il 27/2/2014 (Romeo Rizzi)

#define NDEBUG   // NDEBUG definita nella versione che consegno
#include <cassert>
#ifdef NDEBUG
  #define EVAL
#endif
#include <iostream>
#include <fstream>
#include <vector>

using namespace std;

const int MAXN = 100000;
const int MAXM = 100000;  //Ho preferito ridurre rispetto al bound del testo originale per rimanere
 naturalmente entro lo stack di default sulla mia architettura.

struct edge{
  int el[2];
  bool valid;
  edge(int a,int b){
    el[0]=a;
    el[1]=b;
    valid=true;
  }
};

struct elink{
  int edge_id;
  int order;
  elink(int ed,int ord){
    edge_id=ed;
    order=ord;
  }
};

vector<edge> edges;
vector<vector<elink> > graph;
vector<int> path;


int N, M, X, Y;

void dfs(int el) {
  for(unsigned int i=0; i<graph[el].size(); i++) {
     elink e=graph[el][i];
     if(edges[e.edge_id].valid) {
        edges[e.edge_id].valid=false;
        dfs(edges[e.edge_id].el[e.order]);
     }
  }
  path.push_back(el);
}


int main() {
#ifdef EVAL
        freopen("input.txt","r", stdin);
        freopen("output.txt","w", stdout);
#endif

  cin >> N >> M >> X >> Y;
  X--; Y--;
  graph.reserve(N);
  for(int i=0; i<M; i++) {
    int a, b;
    cin >> a >> b;
    a--; b--;
    graph[a].push_back( elink(edges.size(),1) );
    graph[b].push_back( elink(edges.size(),0) );
    edges.push_back( edge(a,b) );
  }
  dfs(Y);
  assert( (unsigned int)M == path.size()-1 );
  for(unsigned int i=0; i<path.size()-1; i++)
    cout << path[i]+1 << " " << path[i+1]+1 << endl;
  return 0;
```

matita.cpp

```
}
```