Sample Solution to Assignment 2, Problem 2

« Back to Assignments

COURSE HOME
SYLLABUS
CALENDAR
GETTING STARTED
LECTURE NOTES
ASSIGNMENTS
RELATED RESOURCES
DOWNLOAD COURSE MATERIALS

```
/*
PROG: mst
LANG: C++
*/
#include <vector>
#include <queue>
#include <fstream>
#include <iostream>
#include <iomanip>
#include <unordered_map>

class State {
  size_t _node;
  double _dist;
public:
  State( size_t aNode, double aDist ) : _node{aNode}, _dist{aDist} {}
  inline size_t node()const { return _node; }
  inline double dist()const { return _dist; }
};

class AdjacencyList {
  std::vector< std::vector< State>  > _adj;
  AdjacencyList() = delete;
public:
  AdjacencyList( std::istream &input );
  inline size_t size() const { return _adj.size(); }
  inline const std::vector& adj(size_t node )  const { return _adj[node]; }
```

```
  void print();
};

inline bool operator<( const State &a, const State &b ) {
  return a.dist() > b.dist();
}

AdjacencyList::AdjacencyList( std::istream &input ) : _adj{} {
  size_t nNodes; size_t nEdges; input >> nNodes >> nEdges;
  _adj.resize( nNodes );

  for( size_t e = 0; e < nEdges; ++e ) {
    size_t v, w; double weight;
    input >> v >> w >> weight;
    // Add this edge to both the v and w lists
    _adj[v].push_back( State{ w, weight } );
    _adj[w].push_back( State{ v, weight } );
  }
}

void AdjacencyList::print() {
  for( size_t i = 0; i < _adj.size(); ++i ) {
    std::cout << i << ": ";
    for( auto state : _adj[i] ) {
      std::cout << "(" << state.node() << ", " << state.dist() << ") ";
    }
    std::cout << "\n";
  }
}

double prim( const AdjacencyList &adj ) {
  std::unordered_map<int, bool> visited;
  std::priority_queue<State> pq;

  pq.push( State{ 0, 0.0 } );
  double weight = 0.0;

  while( visited.size() < adj.size() ) {
    auto top = pq.top(); pq.pop();

    if( visited.count( top.node() ) == 0 ) {
      visited[top.node()] = true;
      weight += top.dist();
```

```
      for( auto vertex : adj.adj( top.node() ) ) {
        pq.push( vertex );
      }
    }
  }

  return weight;
}

int main() {
  std::ifstream input{ "mst.in" };
  std::ofstream output{ "mst.out" };

  if( input.is_open() ) {
    auto adj = AdjacencyList{ input };
    output << std::fixed << std::setprecision( 8 );
    output << prim( adj ) << "\n";
  } else {
    std::cerr << "Could not open mst.in\n";
    return 1;
  }

  return 0;
}
```

Below is the output using the test data:

```
mst:
 1: OK [0.004 seconds]
 2: OK [0.004 seconds]
 3: OK [0.004 seconds]
 4: OK [0.006 seconds]
 5: OK [0.093 seconds]
 6: OK [0.122 seconds]
 7: OK [0.227 seconds]
 8: OK [0.229 seconds]
 9: OK [0.285 seconds]
10: OK [0.287 seconds]
```

« Back to Assignments

ABOUT MIT OPENCOURSEWARE

MIT OpenCourseWare makes the materials used in the teaching of almost all of MIT's subjects available on the Web, free of charge. With more than 2,400 courses available, OCW is delivering on the promise of open sharing of knowledge. Learn more »

Massachusetts Institute of Technology

OFFICE OF DIGITAL LEARNING

OPEN EDUCATION CONSORTIUM
The Global Network for Open Education

CC BY NC SA