USACO FEB12 Problem 'relocate' Analysis

We first use Dijkstra's algorithm to compute the shortest path from each market to each town. Then for each prospective town x in which Farmer John might build his house, we check all possible K! permutations of the markets that could end up being a feasible daily schedule (checking each one is fast since we have now computed all the relevant market-town shortest path distances). Among all these, we remember the best solution.

```
#include <iostream>
#include <fstream>
#include <queue>
#include <vector>
#include <map>
#include <cstdlib>
#include <cmath>
#include <functional>
#include <cstring>
#include <algorithm>

#define pii pair<int,int>

using namespace std;

int N,M,K;
int markets[5];
int inf = 1 << 29;
vector<pii> graph[10005]; //L then end
int shortest[5][10005]; //shortest path from the ith market to the jth town
bool isMarket[10005]; //is town i a market?

void dijkstra (int start) //from a market
{
  priority_queue <pii, vector<pii>, greater<pii> > pq;
  pq.push(pii(0, markets[start]));

  while(!pq.empty()) //standard heap dijkstra
    {
      int curdist = pq.top().first;
      int curnode = pq.top().second;
      pq.pop();

      if(shortest[start][curnode] <= curdist)
        continue;

      shortest[start][curnode] = curdist;

      for (int i = 0; i < graph[curnode].size(); i++)
        {
          int nextnode = graph[curnode][i].second;
          int nextdist = graph[curnode][i].first + curdist;

          if(nextdist < shortest[start][nextnode])
            pq.push(pii(nextdist, nextnode));
        }
    }
}

int main()
{
  ifstream in ("relocate.in");
  ofstream out ("relocate.out");

  in >> N >> M >> K;
  for (int i = 0; i < N; i++)
    isMarket[i] = false;
  for (int i = 0; i < K; i++)
    {
      in >> markets[i];
      markets[i]--;
      isMarket[markets[i]] = true;
    }
  for (int i = 0; i < M; i++)
    {
      int a,b,L;
      in >> a >> b >> L;
      a--; b--;
      graph[a].push_back(pii(L, b));
      graph[b].push_back(pii(L, a));
    }
  for (int i = 0; i < K; i++)
    {
      for (int j = 0; j < N; j++)
        shortest[i][j] = inf;
      dijkstra(i);
    }

  int best = inf;
  int order[K];
  for (int i = 0; i < K; i++)
    order[i] = i;

  //loop over all permutations in which the K markets are visited
  do{
    int total = inf;
    for (int i = 0; i < N; i++) //choose the farm location to minimize the
                               // sum of the distances from the farm to the
                               // first market and the last market to the farm
      if(!isMarket[i])
        total = min(total, shortest[order[0]][i] + shortest[order[K-1]][i]);

    for (int i = 1; i < K; i++) //add up distances between pairs of markets
      total += shortest[order[i-1]][markets[order[i]]];

    best = min(best, total);
  }
  while(next_permutation(order, order + K));

  out << best << "\n";
  out.close();
  return 0;
}
```