November 08, 2023

Review-1 — Priority Queues using STL

==========

Uniform Cost Search is an algorithm used to move around a directed weighted search space to go from a start node to one of the ending nodes with a minimum cumulative cost.

UNIFORM COST SEARCH

```
function UNIFORM-COST-SEARCH(problem) returns a solution, or failure
  node ← a node with STATE = problem.INITIAL-STATE, PATH-COST = 0
  frontier ← a priority queue ordered by PATH-COST, with node as the only element
  explored ← an empty set
  loop do
    if EMPTY?(frontier) then return failure
    node ← POP(frontier)  /* chooses the lowest-cost node in frontier */
    if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
    add node.STATE to explored
    for each action in problem.ACTIONS(node.STATE) do
      child ← CHILD-NODE(problem, node, action)
      if child.STATE is not in explored or frontier then
        frontier ← INSERT(child, frontier)
      else if child.STATE is in frontier with higher PATH-COST then
        replace that frontier node with child
```

Your task is to find the shortest route using Uniform Cost Search from a given source to the destination described above.

Input

1. The first input line has two integers **n** and **m**: the number of vertices and edges. The cities are numbered **1,2,…,n.**
2. Next line will contain the source and destination node.
3. After this there are **m** lines describing the edges. Each line has three integers **a, b, and c**: an edge begins at node **a**, ends at node **b**, and its price is **c**.

   Note that each edge is unidirectional. You can assume that it is always possible to get from source to Destination.

Output

the price of the cheapest route from a given source to the destination.

Example

Input:

6 7
1 5
1 4 10
1 2 5
2 3 4
3 5 8
2 6 15
4 6 11
6 5 4

Output:

17
(1->2->3->5)