OBJECTIVES:

- Get basic idea of tree traversing using Uniform-Cost Search (UCS) algorithm.
- Implementing UCS algorithm using your preferred programming language.

BACKGROUND STUDY

- Should have prior knowledge on any programming language to implement the algorithm.
- Need knowledge on queue and priority queue.
- Input a graph.
- UCS algorithm that would be covered in theory class and that knowledge will help you here to get the implementation idea.

RECOMMENDED READING

- http://en.wikipedia.org/wiki/Uniform-cost_search
- https://en.wikipedia.org/wiki/Priority_queue
- BOOK

UNIFORM-COST SEARCH

Uniform-cost search (UCS) is a tree search algorithm used for traversing or searching a weighted tree, tree structure, or graph. The search begins at the root node. The search continues by visiting the next node which has the least total cost from the root. Nodes are visited in this manner until a goal state is reached.

Typically, the search algorithm involves expanding nodes by adding all unexpanded neighboring nodes that are connected by directed paths to a priority queue. In the queue, each node is associated with its total path cost from the root, where the least-cost paths are given highest priority. The node at the head of the queue is subsequently expanded, adding the next set of connected nodes with the total path cost from the root to the respective node.

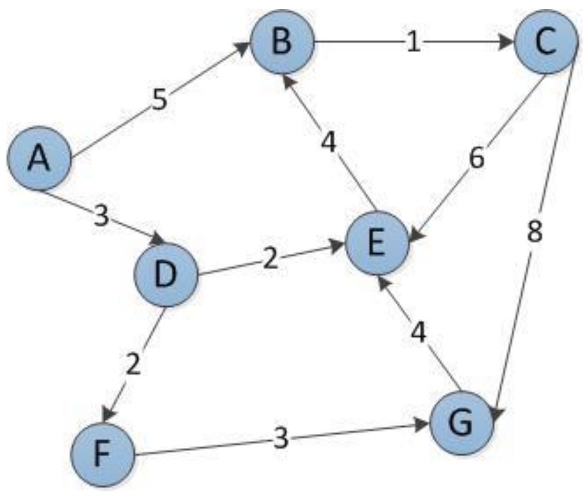

Figure 1: Graph for UCS

Expansion showing the explored set and the frontier (priority queue) for Figure 1:
Start Node: A
Goal Node: G

Step	Frontier: a set of (node, its cost) objects	Expand[*]	Explored: a set of nodes
1	{(A,0)}	A	∅
2	{(D,3),(B,5)}	D	{A}
3	{(B,5),(E,5),(F,5)}	B	{A,D}
4	{(E,5),(F,5),(C,6)}	E	{A,D,B}
5	{(F,5),(C,6)}[**]	F	{A,D,B,E}
6	{(C,6),(G,8)}	C	{A,D,B,E,F}
7	{(G,8)}	G	{A,D,B,E,F,C}
8	∅		

^ * The node chosen to be expanded for the next step.
^ ** B is not added to the frontier because it is found in the explored set.

Pseudocode that can be used to implement BFS algorithm is as following:

```
procedure UniformCostSearch(Graph, root, goal)
  node := root, cost = 0
  frontier := priority queue containing node only
  explored := empty set
  do
    if frontier is empty
      return failure
    node := frontier.pop()
    if node is goal
      return solution
    explored.add(node)
    for each of node's neighbors n
      if n is not in explored
        if n is not in frontier
          frontier.add(n)
        else if n is in frontier with higher cost
          replace existing node with n
```

LABROTORY EXERCISES

1. Implement UCS algorithm using the pseudocode described above in your preferred language.
   Input: a graph G, start node S and a goal node D.
   Output: Goal found or not and the cost.
2. In the 5 by 5 matrix below, the minimal path sum from the top left to the bottom right, by only moving to the right and down, is indicated in bold red and is equal to 2427.

$$\begin{pmatrix} {\color{red}\mathbf{131}} & 673 & 234 & 103 & 18 \\ {\color{red}\mathbf{201}} & {\color{red}\mathbf{96}} & {\color{red}\mathbf{342}} & 965 & 150 \\ 630 & 803 & {\color{red}\mathbf{746}} & {\color{red}\mathbf{422}} & 111 \\ 537 & 699 & 497 & {\color{red}\mathbf{121}} & 956 \\ 805 & 732 & 524 & {\color{red}\mathbf{37}} & {\color{red}\mathbf{331}} \end{pmatrix}$$

   Find the minimal path sum, in matrix.txt (dataset given), a 31K text file containing a 80 by 80 matrix, from the top left to the bottom right by only moving right and down.