AI Homework -1

### Shon Bangale

001327977

sbangale@albany.edu

# Breadth First Search

Breadth first search is a graph traverse and search algorithm for a graph or tree data structure. It begins executing from the tree root node. It then traverses the neighbor as going down in depth of tree one depth level at a time. It uses concepts of backtracking and hence uses Queue data structure to implement BFS.

Complexity: O(V+E)

## Pseudocode

BFS (G, s) //Where G is the graph and s is the source node

let Q be queue.

Q.enqueue( s ) //Inserting s in queue until all its neighbour vertices are marked.

mark s as visited.

while ( Q is not empty)

//Removing that vertex from queue,whose neighbour will be visited now

v = Q.dequeue( )

//processing all the neighbours of v

for all neighbours w of v in Graph G

if w is not visited

Q.enqueue( w ) //Stores w in Q to further visit its neighbor

mark w s visited.

[1] BFS pseudocode

## Implementation

Graph – 1 BFS – Queue implementation adjacency Matrix

Package used: numpy

Output:

Matrix

BFS - Adjacency Matrix: s d e p b c h r q a f g

BFS - > Path - Adjacency Matrix:

s d c f g

Graph – 2 BFS – Queue implementation adjacency Matrix

Queue BFS - Adjacency Matrix :

s d e p b c h r q a f g

Queue-Paths BFS - Adjacency Matrix:

s e r f g

List

Graph – 1 BFS – Queue implementation adjacency List

**The BFS Traversal**

**s d e p b c h r q a f g**

Graph – 2 BFS – Queue implementation adjacency List

**The BFS Traversal**

**s d e p b c h r q a f g**

# Depth First Search

Depth first search a contrary to the above algorithm starting from the root of the tree or a graph traverses the entire depth of a branch at a time.

Complexity: O(V+E)

## pseudocode

DFS(G, u)

u.visited = true

for each v ∈ G.Adj[u]

if v.visited == false

DFS(G,v)

init() {

For each u ∈ G

u.visited = false

For each u ∈ G

DFS(G, u)

}

[2] DFS pseudocode

DFS can be implemented using both Iterative (Stack) and recursion method.

The performance of both are quite like each other. The only difference is ease of development.

## Implementation

Package used numpy

LIST

Recursion

Graph 1

The Path of DFS using List Graph 1

s d b a c f g r e h p q

Graph 2

The Path of DFS using List Graph 2

s d b a c e h p q r f g

Stack

The Path of DFS using List Graph 1

s p q h e r f g c d b a

The Path of DFS using List Graph 2

s p q e r f g c a h d b

Matrix

Recursion

DFS using Reccursion and Matrix G1

s p q h e r f g c d b

Path of DFS

s p q h e r f g

DFS using Reccursion and Matrix G2

spqerfgchdb

Path of DFS

s e r f g

Stack

DFS Stack implimentation path G1:

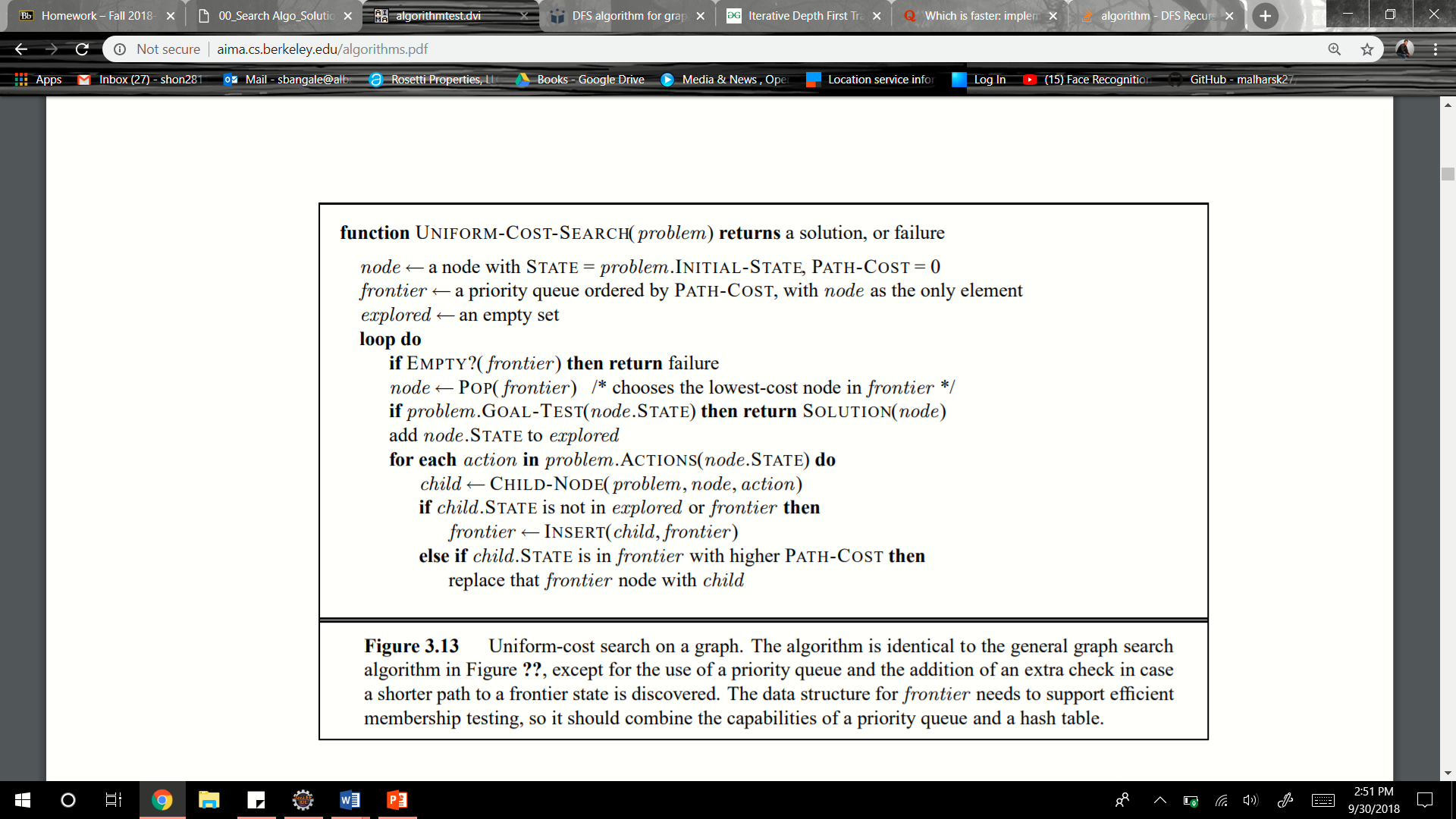
s p q h e r f g

DFS Stack implimentation path G2:

s e r f g

# Uniformed Cost Search

Uniform Cost Search is the best algorithm for a search problem, which does not involve the use of heuristics. It can solve any general graph for optimal cost. Uniform Cost Search as it sounds searches in branches which are the same in cost. Uniform Cost Search again demands the use of a priority queue. [3]



Uniformed cost search Algorithm [4]

Complexity : O(bcost/min\_costedge)

## Implementation

Package: heapq

List

Graph 3:

Vertex List

Node= s Node= p Node= d Node= b Node= e Node= a Node= h Node= r Node= c Node= f Node= e Node= q Node= c Node= f Node= g

Path of UCS

s d e r f g

Graph 4

Vertex List

node = s node = p node = d node = b node = e node = a node = r node = f node = e node = c node = g

Path of UCS

s d e r f g

Matrix

Graph 3

Vertex List

node = s node = p node = d node = b node = e node = h node = a node = r node = c node = e node = f node = q node = c node = f node = g

Path of UCS

s d e r f g

Graph 4

Vertex List

Vistied node = s Vistied node = p Vistied node = d Vistied node = b Vistied node = e Vistied node = a Vistied node = r Vistied node = e Vistied node = f Vistied node = c Vistied node = g

Path of UCS

s d e r f g

# CONCLUSION

I have implemented the following:

The zip folder attached has the source codes of all above outputs

# References

[1] <https://www.hackerearth.com/practice/algorithms/graphs/breadth-first-search/tutorial/>

[2] <https://www.programiz.com/dsa/graph-dfs>

[3] <https://algorithmicthoughts.wordpress.com/2012/12/15/artificial-intelligence-uniform-cost-searchucs/>

[4] <http://aima.cs.berkeley.edu/algorithms.pdf>