AI Homework 2

Lauren Mitchell

Discussed with Taylor Andrews

September 8, 2016

1. BFS is guaranteed to find the shortest path. It is not guaranteed that DFS will ever find the optimal shortest path. When using BFS, once we have visited a node in the graph we now know the shortest path from the vertex to that node. Therefore, by the time we have iterated through the entire graph, we are able to find the shortest path to any point from the vertex.

Given this tree find the optimal path to Node 7. [1,2,5,7]

Current Node: 4

Queue: 6

1. State Space

Successor: 2n, 2n+1

1. Goal state: 11

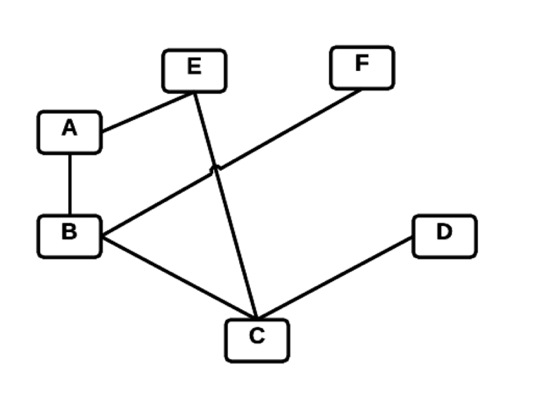
BFS: [1,2,3,4,5,6,7,8,9,10,11]

Assuming that root depth is 0, DLS where n= 3: [1,2,4,8,9,5,10,11]

IDS: [1,1,2,3,1,2,4,5,3,6,7,1,2,4,8,9,5,10,11]

1. The purpose of *visited* node in the DFS search tree is to show what nodes have already been visited and there is no need to “re-iterate” through that portion of the tree.

If there was no visited node, then the search would recurse infinitely.

 [A,B,C,D,C,D,C,D…] it could potentially get stuck going back and forth between C and D infinitely because it will never know that C or D was previously visited.

1. Psuedo code from notes:

DLS(L, goal, depth)

If L.key == goal

Return vertex

If depth == 0

Return cutoff

L.visited = true

Cutoff\_occured = false

For each v in L.adjacent

If(!v.visited)

v.parent = vertex

result = DLS(v, goal, depth --)

if result == cutoff

cutoff\_occured = true

else if result =! Failure

return result

if cutoff\_occured

return cutoff

else

return failure

IterativeDeepening(vertex,goal,maxDepth)

Create empty list L

For x = 0 to maxDepth

Result = DLS(vertex, goal, x)

If result is not failure or cutoff

Return result

Each iteration add to the list L

Return failure

This is essentially a modified breadth first search. Instead of passing in vertex, which is always going to be the root node, you pass in the list L. The last node in list L will be what we are referencing and starting at. We pick up where we left off in the last iteration instead of going back to the root of the graph. We cannot have a single variable be the vertex because there are various places in the graph that we could be left off.

1. False because in Dijkstra’s algorithm is greedy and will often solve paths that are not necessarily shorter. For example, in the example presented to use in class, C is a solved node early in the algorithm. However, the total path distance when using C node is 14, where there are two other paths that are only a total distance of 13.