Ben Pung

Prof Orduz

Advanced Algorithms

22 October 2023

4.3 Graphs

1. Usually, once a vertex is marked in a depth first search, it stays marked. But there are some applications where we want to unmark a vertex. Here is one. Assuming that our graph is a directed acyclic graph, what does this code do?

The code in this question shows that in normal depth first search we mark a point such that we do not visit it again; however, in this version the nodes are marked as unvisited after their subtree (all further edges. In the event that a node/edge is changed, added, updated, or removed, we are able to go back and search through the tree again.

1. Finding the shortest path from vertex A to vertex B is a breadth first search problem. However, pure breadth-first search has an undesirable side effect, namely that it searches in every direction before moving further away from the starting vertex. If you’re looking for a route from here (Waco, TX) to the Rangers Baseball Stadium in Dallas, you’re not really interested in routes that take you to some place in Austin. (Dallas is North of Waco, Austin is South at approximately the same distance.) A better method for finding routes is called Astar. This algorithm computes the distance from the starting vertex to the ending vertex, and organizes the breadth-first queue, not by distance from the starting vertex, but by distance to the ending vertex. Assume that your graph represents a rectangular grid. You are starting at position (0,0) and wish to get to position (16,16). How many vertices will breadth-first search examine before finding (16,16)? How many vertices will Astar examine?

If I search with breadth-first search in a 16x16 grid, each iteration of BFS will search all immediate neighbors next. This means that at 0,0 I will search (1,1) ,(0,1) and (1,0) next in a “wave” like state that does not stray from each origin. This would mean I search a total of 17 \*17 total times before I get to position (16,16) for a total of 289 times. You need a 17x17 matrix if you have 0,0 and 16,16 because this assumes you have 17 rows and 17 columns that is why you get 289.

The Astar method is commonly used in pathfinding such that it uses a cost and heuristic function to ensure we are going in the quickest way possible. Since we know the grid and can define how to move closer to 16,16 I would say the astar method would move closer by going in a diagonal direction of this grid (0,0) -> (1,1) -> (2,2) -> (3,3) …. for total of 16 squares explored total assuming you are already at (0,0).

3. Does the choice of distance function change the number of vertices that Astar will examine, assuming that there are no obstructions?

Assume we start at (0,0) and then end at (16,16) the number of steps from the given formulas are as follows:

D1 = SQRT((16-0)^2 + (16+0)^2) = SQRT(512) = 22.63

D2 = (16-0) + (16-0) = 32

D3 = max((16-0) , (16-0)) = 16

From the formulas above we can see that the algorithms do in fact change the number of steps that would need to be taken.