CS 4102 Advanced Written 2 - Graph

Joseph Lee

April 1, 2021

1

Proof by contradiction:

Assume e = (u,v) does not belong to some minimum spanning tree of G

Hence, there must be at least one shorter path from u to v

$\Rightarrow$ e' = (u,v)' $\leq$ e = (u,v)

This means (u,v)' must be processed by the MST before (u,v)

However, (u,v) is given to be a minimum-weight edge

$\Rightarrow$ Kruskal algorithm must process (u,v) first

$\Rightarrow$ Contradict the assumption

$\Rightarrow$ e=(u,v) must belong to some minimum spanning tree of G

2

1) Redesign the graph:

Given the current nodes represent the position of each robot, I redesign the graph so that each node now store the positions of both robots and call it graph G'=(V',E'). One way that this can be done is by making each node contain the entire graph G, in which the positions of both robots satisfy the requirement that they cannot be on the same or adjacent nodes.The edges represent whether the robots can move from their current positions to the next one without violating the rule of not getting close but still heading towards their respective destinations. The new graph G' now contains all the possible steps (as vertices) from different paths that two robots can use to reach their destinations.

2) Optimize the schedule

Given the redesigned graph, I optimize the schedule by using BFS to find the shortest path from the robots initial positions: node (s1,s2) to theif final destinations: node (d1,d2). Running BFS takes O(V'+E'), in which V' is the number of vertices that represents the current positions of both robots, and E' is the number of edges that represent the next valid steps the two robots can make. Relating the time complexity back to the given graph G = (V,E), one way we can consider the runtime is that in with V steps, if robot 1 gets to move first, it has a maximum number of possible V to be its next step/position. With one vertice taken by robot 1, robot 2 now have a maximum number of next steps to be V-1. Hence, the runtime for this new algorithm that design an optimal schedule for two robots take $O(V(V-1))$ or O(V^2) As the number of robot grows, we will have a runtime of something like this: V(V-1)(V-2)(V-3)...(V-n) with n being the number of robots, or O($\frac{V!}{(V-n)!}$). In other words, as the number of the robots grows, time complexity decreases.

3

Supposed there is an undirected tree G. DFS and BFS will produce a tree, therefore they must contain all the edges of graph G. The base case for this case is n = 1 where BFS and DFS produce the same tree because there are no edges connected to the root. If you add a node to the graph as n + 1, the case still holds that BFS and DFS produce the same tree. If you add one more node to the G as n + 2, it could have a two cases where the first case is that the third node is connected to only one of the node to make a tree or second of it has edges going to both node. The first case is a valid case where BFS and DFS produce the same tree, but the second case is invalid. Therefore, in order for both BFS and DFS to produce the same tree, then the entire graph G was supposed to be already a tree.