CS 515 Take-Home Worksheet #3—Analysis of Graph Algorithms

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name: | Liam Warren | UNH User ID: |  | Total: |  | /100 pts |

Lecture Section: 01

**[50 pts]** Consider a graph laid out like a regular grid, where each vertex is connected to at most 4 other vertices (north, south, east, and/or west). Furthermore, assume that the graph has *n* vertices on each side of the graph and is fully filled-in, so that |V| = O(*n*2). The diagram at right is an example where *n* = 5. Finally, assume that the east-west edges of the graph all have smaller weights than the north-south edges of the graph.

Determine the worst-case running time of Prim’s algorithm on such a graph in terms of *n*, and explain your reasoning for why it is so. Note that because all the east-west edges have smaller weights than the north-south edges, the resulting MST will consist of horizontal lines connected by a single north-south edge on either side. Don’t try to use the general running time in the notes, but instead base your analysis on your knowledge of the algorithm.

The worst case running time of Prim’s algorithm in general is O(|E|log|V|) where |E| is the number of edges and |V| is the number of verticies. This is because in the algorithm must insert each vertex into a priority queue which take O(log(n)) time and process each edge associated with that vertex that isnt already in the tree (which in the worst case will be every edge for every vertex). In this specific case because the north south edge weights will always be larger than the east west edge weights, the algorithm will process each row before moving on to the next, thus elimating 1 edge from each vertex in each row processed after the first. This implies the first row processed will take O(|E|log|v|) time, and being that each vertex has at most 4 edges, each row after the first will take O(3/4|E|log|v|) time. Considering that the graph has n verticies in a row and n rows, the worst-case running time of Prim’s algorithm in this case will be O(4log(n) + 3log(n2- n)) which simplifies to O(3log(n2))

*More on the back…*

**[50 pts]** Determine the worst-case running time of Kruskal’s algorithm in terms of *n* on a regular grid like the one on the other page, except that you can ignore the edge-weight restriction. Explain your reasoning. Don’t try to use the general running time in the notes, but instead base your analysis on your knowledge of the algorithm and the data structures it is using.

Simialarly to Prim’s Algorithm, Kruskal’s algorithm’s running time in general is O(|E|log|V|). This again is because insertion into a priority queue take O(logn) time and in the worst case each edge needs to be processed. In this specific case, the edges in the priority queue will be sorted in such a way that all of the north south edges will be below the east west edges, which leads to the worst case being that all of the edges which connect the final row are the heighest weighted edges. this causes the algorith to process all but n -1 of the edges in the graph because the first edge that connects the final set will complete the MST. Therfore, the worst case running time in this case is O(n2-(n-1)log(n2)) which simplifies to O(n2log(n2)), so this set represents a worst-case for Kruskuls algorithm.