Problem 1. Dijkstra’s Algorithm

(a) Weiss, problem 9.5(a). Use Dijkstra’s algorithm and show the results of the algorithm in the form used in lecture — a table showing for each vertex its known distance from the starting vertex and its predecessor vertex on the path. Also show the order in which the vertices are added to the “cloud” of known vertices as the algorithm progresses.

(b) What is the worst-case complexity of Dijkstra’s algorithm when implemented with $d$-heaps?

(c) Suppose $|E| = |V|$. Are you better off implementing Dijkstra’s algorithm with a $d$-heap, compared to using a binary heap or no heap at all? Justify your answer.

(d) Suppose $|E| = |V^2|$. Are you better off implementing Dijkstra’s algorithm with a $d$-heap, compared to using a binary heap or no heap at all? Justify your answer.

(e) Is there any value of $|E|$ between $|V|$ and $|V^2|$ and a value of $d$ for which implementing Dijkstra’s algorithm with a $d$-heap performs better than BOTH binary heaps or no heaps at all? Explain your answer.

Problem 2. Minimum Spanning Trees

(a) Weiss, problem 9.15(a). For Prim’s algorithm, start with vertex $A$, show the resulting table (see Table 9.55 as an example), and indicate the order in which vertices are added. For Kruskal’s algorithm, produce a table, similar to Table 9.56. Ties may be broken arbitrarily.

(b) Weiss, problem 9.15(b).

(c) Extra Credit. Suppose all the edge weights in a connected, undirected graph are integers between 1 and $|E|$. How could you use this fact to make a more efficient version of Kruskal’s algorithm, and what would be its worst case complexity?