Some sample problems for Exam 2
Exam date: **Wednesday**, November 6, 8:15 - 9:45 pm
Sections A and D: Kildee 0125
Section C: Gerdin 1148

- In general you may use the following algorithms and their time bounds as a "black box" on the exam (that is, unless modifications are needed, you do not need to write the code nor derive runtime for them): sorting, breadth-first search, depth-first search, Prim's algorithm, Kruskal's algorithm, Dijkstra's algorithm, topological sort.

  However, if you modify any of these methods, then you must write a complete description of the modified method. Merely stating the modification does not suffice.

- For any problem or sub-problem, if you write "*Do not grade*" and nothing else, you will receive 20% credit for it.

- For all algorithm problems, part of the grade depends on the efficiency.

- Don't forget to review the problems from homeworks 4 and 5 as well.

1. Figure 1 shows a DAG. Write all possible topological orderings of this graph.

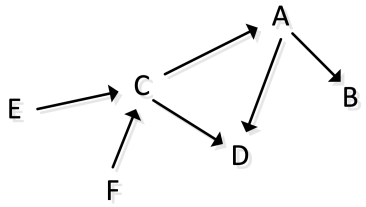


Figure 1: A DAG.

2. Trace execution of Dijkstra's algorithm using the worksheet on the last page.

3. Give an algorithm that gets a DAG G as input and outputs the number of possible topological orderings.

4. Write a pseudocode implementation of Dijkstra's algorithm using a priority queue that does *not* have a `ChangeKey` operation as described in the text. Can you still do this in $O(m \log n)$?

5. Suppose we run Prim's algorithm on a graph G with s as starting node and let T be the MST constructed. Is T a a shortest path tree? I.e, is the length of the path from x to any vertex v in T equal to the length of the shortest path from x to v in G?

6. Section 3.5 from the text provides an overview of an algorithm to compute all connected components of an undirected graph and claims that the algorithm can be made to run in $O(m+n)$ time. However, the algorithm has the following step: "find a node that has not been visited by the search from s". At first glance, it seems that this step will take $O(n)$ time and thus the algorithm should be $O(n(m+n))$. Provide details showing how the algorithm can be made to run in $O(m+n)$ time.

7. Solve the *single-destination shortest path* problem: Given a directed graph G with positive edge weights, and a vertex s, find the shortest path from every vertex u to s.

8. What is the Big-O complexity of determining whether a directed graph has a cycle? What about an undirected graph?

9. Let $G = (V, E)$ be an undirected graph in which the edge weights are not necessarily distinct, and let S be any subset of V such that S is nonempty and $S \neq V$. Let $e(x, y)$ be an edge of minimal weight such that $x \in S$ and $y \in V - S$. Prove that there exists a MST that contains edge e.

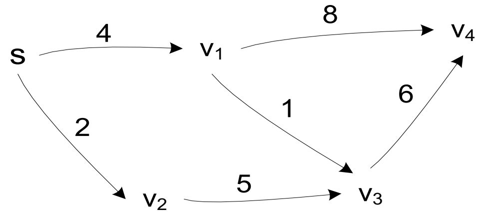
10. Let G be an undirected weighted graph. Let T be its MST and Q be its shortest path tree (from a node s). Suppose we increase the weight of every edge by 1. Is T still an MST for the new graph? Is Q still the shortest path tree for the new graph? Prove or give a counterexample.

11. Prove the correctness of Kruskal's algorithm using an exchange argument.

12. let $m_1, m_2, \cdots, m_n$ be distinct non-integers on the number line, in increasing order. Your goal is to color all of them blue. You have magical blue pens with the following property: When you place the pen at co-ordinate x, all the points in the range $[x-5, x+5]$ turn blue. A pen can be used only once. Give an algorithm to color all the points using as few pens as possible. Prove the correctness of the algorithm and derive the run-time.

13. You have n tasks to complete, where each task takes exactly one unit of time. Each task i comes with a deadline time d_i, and a value r_i, such that if you complete it by d_i, you will be rewarded r_i dollars. Give a greedy algorithm that picks tasks so that it will maximize the rewards. Prove the correctness using an exchange argument.

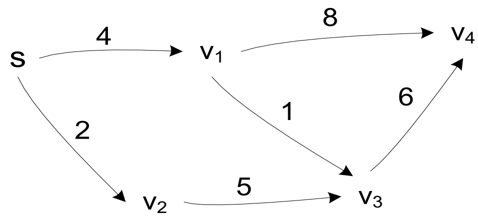
Execute Dijkstra's algorithm on the directed graph below. At each step **circle** the elements of the set S (nodes for which a shortest path is known), and show the predecessors and distances. (Distances that are not filled in are assumed to be "infinity".)



pred

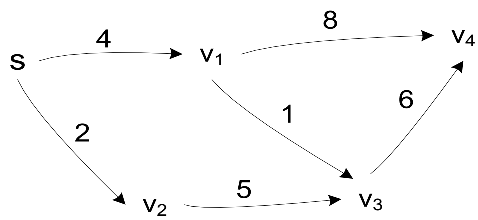
dist

s	0



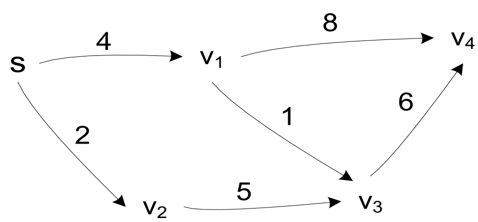
pred

dist



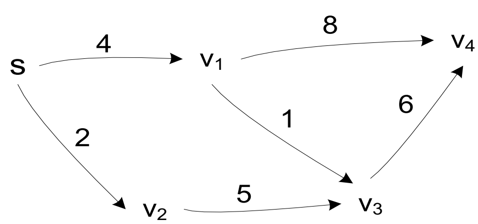
pred

dist



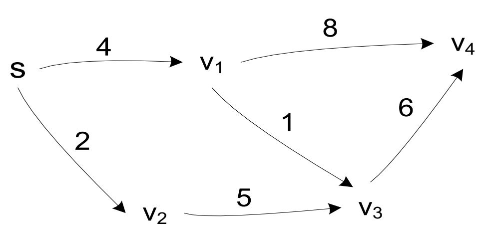
pred

dist



pred

dist



pred

dist