University of Washington October 30, 2020
Department of Computer Science and Engineering
CSE 417, Autumn 2020

Homework 5, Due Friday, November 6, 1:30 pm, 2020

Turn in instructions: Electronic submission on GradeScope. Submit as a PDF, with each problem on a separate page. While we have encouraged typeset solutions (e.g., Word or LaTex) in the homework, it is probably easiest to hand write solutions for the practice midterm exam (as the midterm was designed for pen and paper).

This homework assignment has two parts - a programming component (Dijkstra!) and a sample midterm. For the sample midterm, you are strongly encouraged to take the midterm under exam conditions - closed book, 50 minutes, to see how you perform on the questions as an assessment. After you have completed the practice midterm, you may take a second pass over the exam with additional time and resources to write up solutions to turn in.

The midterm questions are questions 1-7, and programming questions are 8-10. A separate copy of the midterm is also available for download to use in a sample exam setting. Problem number 6 requires material from Friday's lecture. (The schedule slipped by a lecture.)

Problem 1 Graph Theory (5 points):

a) *True or false*: Let $G = (V, E)$ be an undirected graph. If G is a tree, then G is bipartite. Justify your answer[1].

b) *True or false*: Let $G = (V, E)$ be a directed graph with n vertices and m edges. It is possible to determine if G has a cycle in $O(n + m)$ time. Justify your answer.

Problem 2 Stable Matching (5 points):

Step through the Gale-Shapely stable matching algorithm on the instance below. (You may choose the proposals in any order.) The preference lists are:

$$M = \begin{bmatrix} m_1: & w_1 & w_2 & w_3 & w_4 \\ m_2: & w_1 & w_3 & w_4 & w_2 \\ m_3: & w_2 & w_1 & w_3 & w_4 \\ m_4: & w_2 & w_1 & w_3 & w_4 \end{bmatrix}$$

$$W = \begin{bmatrix} w_1: & m_3 & m_4 & m_1 & m_2 \\ w_2: & m_1 & m_2 & m_4 & m_3 \\ w_3: & m_3 & m_4 & m_1 & m_2 \\ w_4: & m_1 & m_2 & m_3 & m_4 \end{bmatrix}$$

[1] "Justify" means give a short and convincing explanation. Depending on the situation, justifications can involve counter examples, or cite results established in the text or in lecture.

Fill in the following table to trace the algorithm. The first two rows are given.

Round	Proposal	Result	Current Matching
0			$(m_1, *), (m_2, *), (m_3, *), (m_4, *)$
1	m_1 proposes to w_1	w_1 accepts m_1	$(m_1, w_1), (m_2, *), (m_3, *), (m_4, *)$
2	m_2 proposes to w_1	w_1 rejects m_2	$(m_1, w_1), (m_2, *), (m_3, *), (m_4, *)$
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			

Problem 3 Shortest Cycle (5 points):

Let $G = (V, E)$ be an undirected graph. Let $e = \{u, v\}$ be an edge in G. Give an $O(n + m)$ time algorithm that finds the shortest cycle in G which contains the edge e. Explain why your algorithm is correct.

Problem 4 Connected Components (5 points):

Suppose $G = (V, E)$ is an undirected graph with n vertices and n edges. (Note: G is not allowed to have self loops or parallel edges.)

a) What is the minimum number of connected components that G can have? Justify your answer.

b) What is the maximum number of connected components that G can have? Justify your answer.

Problem 5 Interval Scheduling (5 points):

The input for an interval scheduling problem is a set of intervals $I = \{i_1, \ldots, i_n\}$ where i_k has start time s_k, and finish time f_k. The problem is to find a set of non-overlapping intervals that satisfies a given criteria.

a) Suppose that you want to maximize the total length of the selected intervals. *True or false*: The greedy algorithm based on selecting intervals in order of decreasing length finds an optimal solution. Justify your answer.

b) Suppose that all intervals have the same length, and you want to maximize the total length of the selected intervals. *True or false*: The greedy algorithm based on selecting intervals in order of increasing start time finds an optimal solution. Justify your answer.

Problem 6 Recurrences (5 points):

Solve the following recurrences by unrolling the recursion tree. Express your answers as $O(f(n))$.

a)

$$T(n) = \begin{cases} 5T(\frac{n}{3}) + n & \text{if } n > 1 \\ 1 & \text{if } n \leq 1 \end{cases}$$

b)

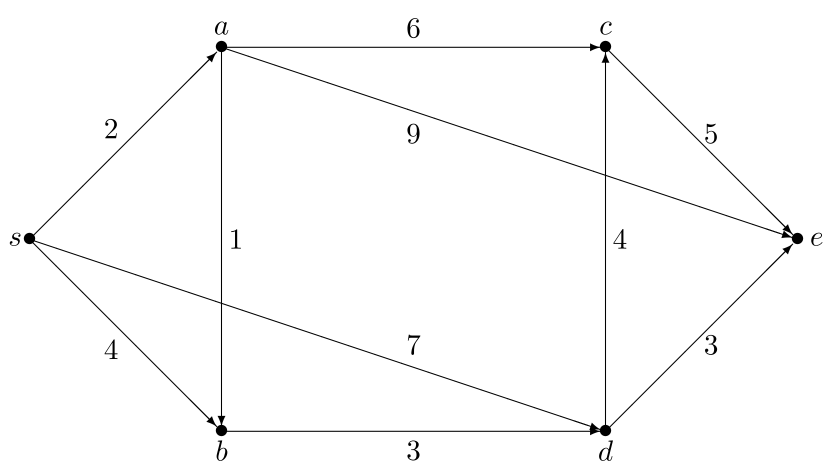
$$T(n) = \begin{cases} T(\frac{4n}{5}) + n & \text{if } n > 1 \\ 1 & \text{if } n \leq 1 \end{cases}$$

c)

$$T(n) = \begin{cases} 16T(\frac{n}{4}) + n^2 & \text{if } n > 1 \\ 1 & \text{if } n \leq 1 \end{cases}$$

Problem 7 Dijkstra's Algorithm (5 points):

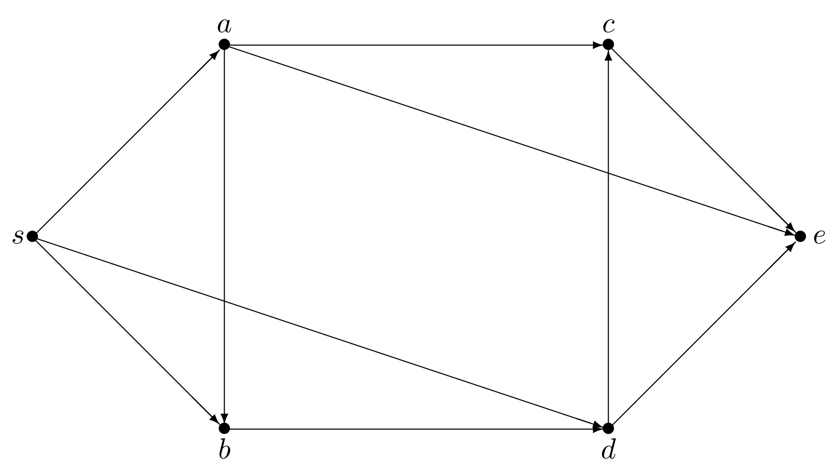
Use the following graph to simulate versions of Dijkstra's algorithm in parts a) and c) starting from the vertex s.



a) Simulate Dijkstra's shortest path algorithm on the graph above by filling in the table. The entries should contain the preliminary distance values.

Round	Vertex	s	a	b	c	d	e
1							
2							
3							
4							
5							
6							

b) Draw the back edges found by your simulation of Dijkstra's algorithm.



Problem 8 Grid graph generator (5 points):

The purpose of this problem is to construct a random graph generator that should be interesting for shortest path's algorithms. An $n \times n$ directed grid graph has the vertex set

$$V = \{\langle i, j \rangle \mid 1 \leq i \leq n \text{ and } 1 \leq j \leq n\}$$

and edge set

$$E = \{(\langle i, j \rangle, \langle i{+}1, j \rangle) \mid 1 \leq i \leq n{-}1 \text{ and } 1 \leq j \leq n\} \cup \{(\langle i, j \rangle, \langle i, j{+}1 \rangle) \mid 1 \leq i \leq n \text{ and } 1 \leq j \leq n{-}1\}.$$

The base graph, for a given n is an $n \times n$ grid graphs. The edge costs are random real (or floating point) numbers x chosen uniformly in the range $0 \leq x < 1$.

For this problem, create the generator, and print an example random grid graph for $n = 4$. It is sufficient to print out a list of edges (and costs) that the graph has.

Problem 9 Dijkstra's Shortest Path Algorithm Implementation (10 points):

Implement Dijkstra's Shortest Path algorithm and run the algorithm on the grid graph from problem 8. The starting vertex is $s = \langle 1, 1 \rangle$ and you are interested in finding the distance to $t = \langle n, n \rangle$. For running your algorithm, you may use $n = 100$ (although you could probably use a much larger value of n, maybe as large as $n = 10,000$). Run your algorithm for a number of runs (say 10) to compute the average distance between s and t. Before you run your algorithm, you may want to estimate what the distance should be.

Problem 10 Dijkstra's Bottleneck Path Algorithm (10 points):

Same as problem 9, but implement the algorithm to compute the bottleneck path distance.