University of Washington
Department of Computer Science and Engineering
CSE 417, Winter 2020

February 4, 2020

Old midterm questions

Problem 1. Stable Marriage (10 points):

Show that the Gale-Shipley Stable Marriage algorithm can take $\Theta(n^2)$ steps with appropriate choice of preference lists. Give preference lists and an ordering of the proposals that require $\Theta(n^2)$ steps. Explain why your example achieves the bound.

Hint: This can be done with all of the M's having the same preference lists, and all of the W's having the same preference lists.

Problem 2. Big Oh (10 points):

Let q, r, and s be positive constants. Prove that $qn^2 + rn + s$ is $O(n^2)$ using the formal definition of $O(\cdot)$.

Big $O(\cdot)$ definition: $f(n)$ is $O(g(n))$ if there exists $c > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$, $f(n) \leq cg(n)$.

Problem 3. True or False (30 points):

Determine if the following statements are true or false. Provide a short justification for each answer.

a) *True or false:* If G is a directed graph on n vertices where every vertex has out degree at least two, then G has a cycle. Justify your answer.

b) *True or false:* If G is a directed graph on n vertices with at least $2n$ edges, then G has a cycle. Justify your answer.

c) *True or false:* If G is a directed graph on n vertices, with distinct vertices r and s, where there is a path from r to every vertex in the graph, and there is a path from s to every vertex in the graph, then there is a cycle in the graph. Justify your answer.

d) *True or false:* If G is an undirected graph with edge weights, and edge e has weight strictly greater than any other edge in the graph, then e cannot be in a minimum spanning tree for G. Justify your answer.

e) *True or false:* If G is an undirected graph with edge weights, and edge e has weight strictly less than any other edge in the graph, then e must be in every minimum spanning tree for G. Justify your answer.

f) *True or false:* If G is a undirected graph on n vertices with more than $n/2$ connected components, then at least one of the connected components is an isolated vertex. Justify your answer.

Problem 4. Minimum Weight Branching (10 points):

A branching is a rooted subtree in a directed graph where there is a path from the root r to every vertex in the graph. The *minimum branching problem* is: given a directed graph with weights on the edges and a specified vertex r, find a branching of minimum weight rooted at r.

Show that Dijkstra's shortest paths algorithm *does not* solve this problem. Specifically, give a graph where the shortest paths found by Dijkstra's algorithm do not form a minimum weight branching.

Problem 5. One-Two Knapsack Problem (20 points):

The *Knapsack Problem* is: Given a collection of items $I = \{i_1, \ldots, i_n\}$ and an integer K where each item i_j has a weight w_j and a value v_j, find a subset of the items with weight at most K which maximizes the total value of the set. More formally, we want to find a subset $S \subseteq I$ such that $\sum_{i_k \in S} w_k \leq K$ and $\sum_{i_k \in S} v_k$ is as large as possible.

We define the *density* of d_j of item i_j to be $d_j = v_j/w_j$. A natural greedy algorithm for the knapsack problem is to consider the items in order of decreasing density, and place each item into the knapsack if there is still sufficient space for the item.

For this problem, we restrict the weights of the items to be either 1 or 2. For convenience, we assume the capacity K of the knapsack is an even number.

a) Given an example that shows that the greedy algorithm based on sorting items by density does not necessarily give an optimal solution, even if the weights are restricted to 1 and 2.

b) Describe an efficient algorithm that finds an optimal solution to the knapsack problem when the weights are restricted to 1 and 2.

c) Provide a justification that your algorithm is correct.

Problem 6 (10 points):

Consider the stable matching problem.

a) Show that it is possible to have a *last-choice* match: There exists an instance of the problem with a stable matching M that has m matched with w, where w is m's last choice, and m is w's last choice.

b) Is it possible for a stable matching to have two *last-choice* matches: could a stable matching M have m_1 matched with w_1 where m_1 is w_1's last choice and w_1 is m_1's last choice, and m_2 matched with w_2 where m_2 is w_2's last choice and w_2 is m_2's last choice? Justify your answer.

Problem 7 (10 points):

Show that

$$\sum_{k=0}^{\log n} 4^k$$

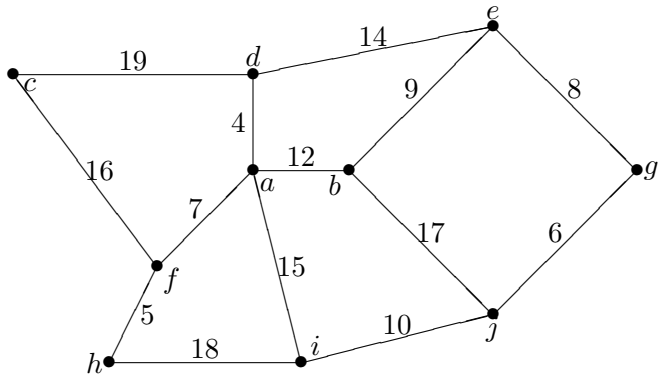
is $O(n^2)$.

Problem 8 (10 points):
Let $G = (V, E)$ be an undirected graph.

a) True or false: If G is a tree, then G is bipartite. Justify your answer.

b) True or false: If G is not bipartite, then the shortest cycle in G has odd length. Justify your answer.

Problem 9 (10 points):
Consider the following undirected graph G.



a) Use the Edge Inclusion Lemma to argue that the edge (a, b) is in every Minimum Spanning Tree of G.

b) Use the Edge Exclusion Lemma to argue that the edge (a, i) is never in a Minimum Spanning Tree of G.

Problem 10 (10 points):
The knapsack problem is: Given a collection of items $I = \{i_1, \ldots, i_n\}$ and an integer K where each item i_j has a weight w_j and a value v_j find a subset of the items which has weight at most K and maximizes the total value in the set. More formally, we want to find a subset $S \subseteq I$ such that $\sum_{i_k \in S} w_k \leq K$ and $\sum_{i_k \in S} v_k$ is as large as possible.
Suppose that the items are sorted in decreasing order of value, so that $v_i \geq v_{i+1}$. A simple greedy algorithm for the problem is:

```
CurrWeight := 0;
Sack := ∅;
for j := 1 to n
  if CurrWeight + w_j ≤ K then
    Sack := Sack ∪ {i_j}
    CurrWeight := CurrWeight + w_j
```

a) Show that the greedy algorithm does not necessarily find the maximum value collection of items that can be placed in the knapsack.

b) Prove that if all weights are the same, then the greedy algorithm finds the maximum value set. (For convenience, you may assume that each item has weight 1).

Problem 11 (10 points):

Give solutions to the following recurrences. Justify your answers.

a)

$$T(n) = \begin{cases} 2T(\frac{n}{3}) + n & \text{if } n > 1 \\ 1 & \text{if } n \leq 1 \end{cases}$$

b)

$$T(n) = \begin{cases} 8T(\frac{n}{2}) + n^3 & \text{if } n > 1 \\ 0 & \text{if } n \leq 1 \end{cases}$$

Problem 13 (10 points):

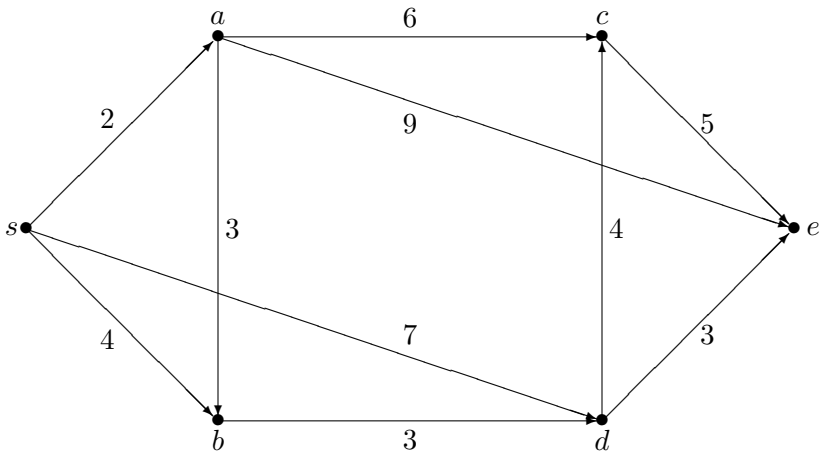
A k-wise merge takes as input k sorted arrays, and constructs a single sorted array containing all of the elements of the input arrays.

a) Describe an efficient divide and conquer algorithm *MultiMerge*$(k, A_1, \ldots, A_k)$ which computes a k-wise merge of its input arrays.

b) What is the run time of your algorithm with input of k arrays of length n. Justify your answer.

Problem 1. Dijkstra's (10 points):

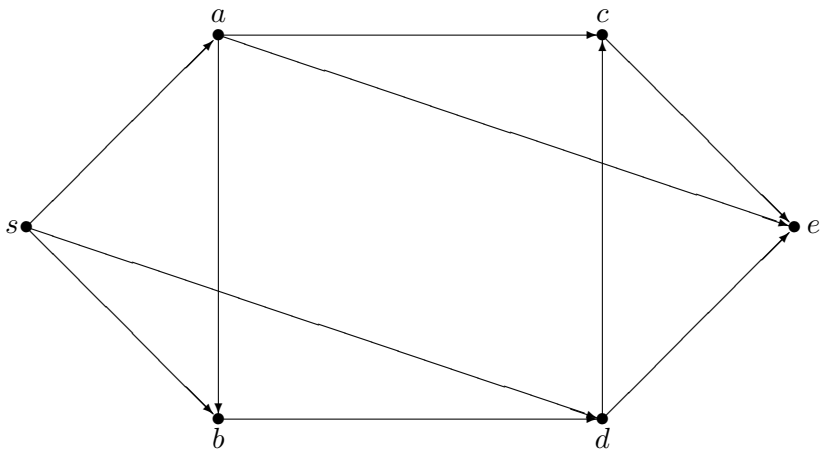
Use the following graph to simulate versions of Dijkstra's algorithm in parts a) and c) starting from the vertex s.

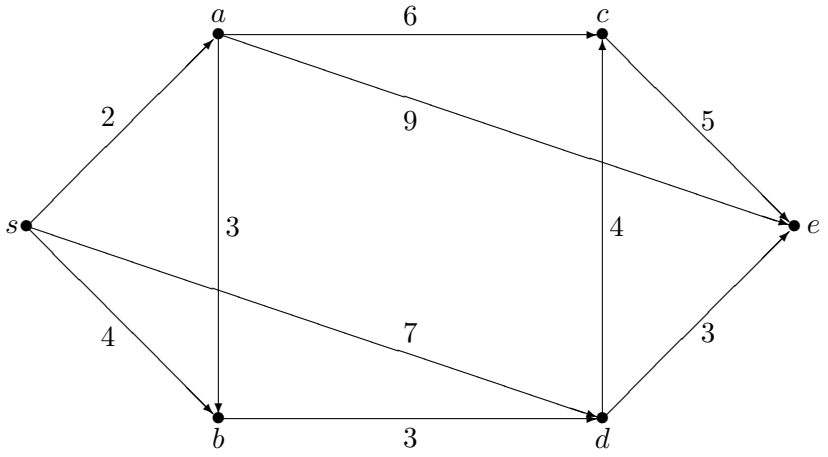


a) Simulate Dijkstra's shortest path algorithm on the graph above by filling in the table. The entries should contain the preliminary distance values.

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.

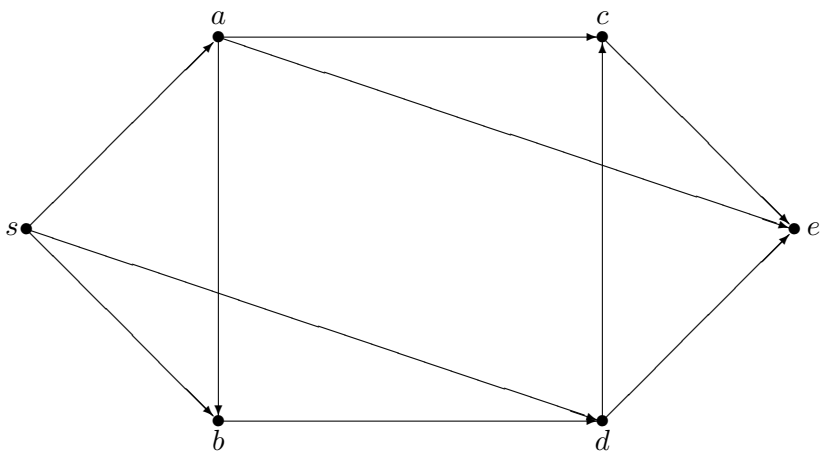




c) Simulate Dijkstra's bottleneck path algorithm on the graph above by filling in the table. The entries should contain the preliminary distance values.

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

d) Draw the back edges found by your simulation of Dijkstra's bottleneck path algorithm.



Problem 14. Fun with Big-Oh (10 points):

a) Order the following functions in increasing order by their growth rate:

1. $(\log n)^{\log n}$
2. n^4
3. $n^3 + n^5$
4. $2^{\sqrt{\log n}}$
5. $(0.01)^n$
6. $2^{n/10}$

b) Is $n^2 \in \Theta(3n^3 + 2n)$? Explain.

c) Is an algorithm with run time $O(n!)$ ever preferable to an algorithm with runtime $O(n)$? Explain.

Problem 15. Minimum Spanning Tree (10 points):

Let $G = (V, E)$ be a connected, undirected graph with edge weights. The edge weights are not necessarily distinct (e.g., the graph may have two or more edges of the same weight).

a) Let e be a minimum weight edge in E. Prove that e is not necessarily in every minimum spanning tree for G.

b) Let e be a minimum weight edge in E. Prove that e is in some minimum spanning tree for G.

Problem 16. Starbucks Placement (10 points):

Washington State law requires that there is a Starbucks within 15 miles when driving on the freeway.[1] The purpose of this problem is to design an algorithm that Starbucks can use to place it's stores along a freeway to ensure that all points are within a fixed distance of a store. The stores can only be placed at off ramps, so there is a list of possible locations given as an increasing sequence of integers. The different directions of the freeway are considered separately, and no back tracking is allowed to reach a store. The problem is: Given a set of integers $A = \{a_1, a_2, \ldots a_n\}$ in increasing order, find a subset S of A which is as small as possible, such that for every $x \in [0, K]$, there is an $a \in S$ with $a - L \leq x \leq a$ where K is the length of the freeway, and L is the maximum allowed distance from a Starbucks.

a) Give a greedy algorithm that finds a minimum sized set of locations that guarantees every point is with distance L of a Starbucks.

b) Prove that the first item that your greedy algorithm selects is a member of some optimal solution to the problem.

Problem 17. Scheduling (10 points):

Let G be the precedence graph (prerequisite graph) for the courses in the Computer Science major. Describe an algorithm for determining the minimum number of quarters to complete the major. (Design the algorithm for over achievers with no bound on the number of courses that can be taken per quarter[2]. Also, assume that every course is offered every quarter.) Justify that your algorithm is correct. You do not need to give the runtime for your algorithm (but it should be a polynomial time algorithm.)

Problem 18 Recurrences (10 points):

Give solutions to the following recurrences. Justify your answers.

a)

$$T(n) = \begin{cases} 4T(\frac{n}{3}) + n & \text{if } n > 1 \\ 1 & \text{if } n \leq 1 \end{cases}$$

b)

$$T(n) = \begin{cases} 25T(\frac{n}{5}) + n^2 & \text{if } n > 1 \\ 1 & \text{if } n \leq 1 \end{cases}$$

[1] This is made up.

[2] This problem is hard if a bound k is put on the number of courses. No efficient algorithm is known for $k = 3$.