University of Washington
Department of Computer Science and Engineering
CSE 417, Winter 2023

December 1, 2023

CSE 417, Final Exam, March 13, 2023

NAME: ____________________________ UW ID: ____________________________

Instructions:

- Closed book, closed notes, no calculators
- Time limit: One hour and fifty minutes
- Answer the problems on the exam paper.
- If you need extra space use the back of a page
- Problems are not of equal difficulty, if you get stuck on a problem, move on.
- "Justify your answer" 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.
- The formula for the sum of the geometric series, for $r \neq 0$ is:

$$\sum_{j=0}^{n} r^j = \frac{r^{(n+1)} - 1}{r - 1} = \frac{1 - r^{(n+1)}}{1 - r}$$

1	/15
2	/12
3	/15
4	/15
5	/15
6	/20
7	/15
8	/13
Total	/120

Notes: For all problems involving graphs, the graph is $G = (V, E)$ with $|V| = n$ and $|E| = m$. You may use common algorithms (e.g., algorithms presented in class) as subroutines without writing them out. You should provide a short justification of why your algorithms work.

Problem 1 (15 points):

a) True or false: For an undirected graph G, if G is a tree, then G is bipartite. Justify your answer.

b) True or false: For a directed graph G with n vertices, if G has at least n edges, then G has a cycle. Justify your answer.

c) True or false: For a directed graph G, if every vertex in G has out degree at least one, then G has a cycle. Justify your answer.

Problem 2 (12 points) Short Answer:

a) What was Steve Cook's role in the development of NP-completeness?

b) Yes, no, or maybe: is Undirected Graph Connectivity NP-Complete? (Undirected Graph Connectivity is: given an undirected graph $G = (V, E)$, determine if there is an undirected path between every pair of vertices.) Justify your answer.

c) What is the run time of the Topological Sort Algorithm.

d) Given a graph G, vertices s and t, and an integer K, is there a polynomial time algorithm to determine if there is a shortest path between vertices s and t that has fewer than K edges?

Problem 3 (15 points) Short Answer:

a) What are the two central ideas involved in Dynamic Programming?

b) How do you determine if a directed graph has a cycle?

c) What is the best case runtime for the stable marriage algorithm for an instance with $|M| = n$ and $|W| = n$, when measured in terms of the number of proposals. Explain.

d) Explain how the Bellman-Ford Algorithm can be applied to currency trading.

e) Give a satisfying assignment to the following boolean formula (we use the notation $\overline{x}$ for $\neg x$ or "not x"):

$$(x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3} \vee \overline{x_4}) \wedge (\overline{x_1} \vee x_3 \vee \overline{x_4}) \wedge (x_1 \vee x_2 \vee \overline{x_4}) \wedge (\overline{x_2} \vee x_3 \vee x_4)$$

Problem 4 (15 points) Unit Size Knapsack:

The Knapsack problem is: Given a set of n items $\{1, 2, \ldots, n\}$ with non-negative weights w_i (for $i = 1, \ldots, n$), and non-negative values v_i (for $i = 1, \ldots, n$), and a weight bound W, find a set of items $S \subseteq \{1, 2, \ldots, n\}$ such that $\sum_{i \in S} w_i = W$ that maximizes the value of the items in the set, i.e., that maximizes $\sum_{i \in S} v_i$. The Unit Size Knapsack restricts the problem so that all of the weights are one, i.e., $w_i = 1$ (for $i = 1, \ldots, n$).

a) Describe a simple greedy algorithm for the Unit Size Knapsack problem. Your algorithm should not be based on Dynamic Programming, and should have runtime $O(n \log n)$ or better.

b) Argue that your algorithm correctly finds an optimal solution. (You may assume that the values are all distinct.)

Problem 5 (15 points) Recurrences:

Give solutions to the following recurrences. Justify your answers by unrolling the recursion tree.

a)

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

b)

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

b)

$$T(n) = \begin{cases} 4T(\frac{n}{2}) + n^3 & \text{if } n > 1 \\ 1 & \text{if } n = 1 \end{cases}$$

Problem 6 (20 points) Highest Value Longest Common Sequence:

The problem will consider strings over the alphabet $\{x_1, x_2, \ldots, x_K\}$ where character x_i has a non-negative integer value v_{x_i}. The value of a string $S = s_1 s_2 \cdots s_n$ is the sum of the character values, i.e., $Val(S) = \sum_{i=1}^{n} v_{s_i}$. The Highest Value Longest Common Sequence (HVLCS) problem is: Given strings $A = a_1 a_2 \cdots a_n$ and $B = b_1 b_2 \cdots b_m$, find a sequence $C = c_1 c_2 \cdots c_k$ such that C is a subsequence of A and B that maximizes $Val(C)$.

a) Consider strings over $\{a, b, c\}$ with $v_a = 2, v_b = 4$, and $v_c = 5$, determine a HVLCS of the strings $A = aacb$ and $B = caab$ and give the value of the HVLCS.

b) Give a recurrence that is the basis of a dynamic programming algorithm to compute the HVLCS of strings $A = a_1 a_2 \cdots a_n$ and $B = b_1 b_2 \cdots b_m$ over the alphabet $\{x_1, x_2, \ldots, x_K\}$ where character x_i has a non-negative integer value v_{x_i}. You must provide the appropriate base cases, and explain why your recurrence is correct.

c) Give pseudocode of an algorithm to compute the length of the HVLCS of given strings A and B. (You do not need to compute the actual string, just determine its value.) What is the runtime of your algorithm?

Problem 7 (15 points) Knapsack Problem:

The Knapsack problem is: Given a set of n items $\{1, 2, \ldots, n\}$ with non-negative weights w_i (for $i = 1, \ldots, n$), and non-negative values v_i (for $i = 1, \ldots, n$), a weight bound W, and target value K, determine if there is a set of items $S \subseteq \{1, 2, \ldots, n\}$ such that $\sum_{i \in S} w_i = W$ and $\sum_{i \in S} v_i \geq K$.

a) Show that the Knapsack problem is in NP.

b) Give a polynomial time reduction from Subset-Sum to Knapsack to prove that Knapsack is NP-Complete.

For this problem, the Subset-Sum problem is: Given a set of n items $\{1, 2, \ldots, n\}$ with non-negative weights w_i (for $i = 1, \ldots, n$) a weight bound W, determine if there is a set of items $S \subseteq \{1, 2, \ldots, n\}$ such that $\sum_{i \in S} w_i = W$. You may assume that this problem has been shown to be NP-Complete.

Problem 8 (13 points):

Let G be a directed acyclic graph, with a distinguished node s. Describe an $O(n + m)$ algorithm to compute, for each vertex v, the number of paths from s to v.