Missouri University of Science & Technology	Department of Computer Science
Spring 2022	**CS 2500: Algorithms**

Sample Questions for Final Exam

Name: *Sid Nadendla* **Email:** *nadendla@mst.edu*

Final exam will be designed as a 2-hour long in-class exam. You may have 6-7 questions to solve, depending on the size of the questions. This handout presents a few review questions for the final examination on May 11, 2022.

Section 1 Asymptotic Notation

1. Definitions for Θ, O, Ω, o and ω notations.
2. Prove that $\sum_{i=1}^{n} i = \Theta(n^2)$.
3. If $f_1(n) = \Theta(g_1(n))$ and $f_2(n) = \Theta(g_2(n))$, prove that
$$f(n) = f_1(n) + f_2(n) = \Theta(g_1(n) + g_2(n)).$$

Section 2 Recursion

1. Using substitution method, prove that the recurrence relation $f(n) = 2f(n/2) + n$ results in $f(n) = O(n \log n)$.
2. Solve $T(n) = \sqrt{n}\, T(\sqrt{n}) + n$ using recursion trees.

Section 3 Divide & Conquer

1. Our goal is to find both maximum and minimum elements of a given input array A of size n efficiently. A naive approach is to find the maximum using $n - 1$ comparisons and then find the minimum element over the remaining entries using $n - 2$ comparisons. Therefore, this algorithm takes about $2n - 3$ comparisons. If you were to adopt a divide-and-conquer approach, what is the order of improvement in terms of the number of comparisons?
2. The problem is to compute the product of two n-bit input arrays A and B, where n is significantly larger than the register size in your processor (e.g. Consider the product of 256-bit arrays over a 64-bit processor). In such a case, it is infeasible to run the traditional method for finding the product of two arrays. How would you solve this problem?

Section 4 Comparison Sorting

1. Write the pseudocode for INSERTION-SORT and prove its correctness.
2. Given two sorted input arrays A and B, how can we merge them into a sorted array? Write the pseudocode for your approach and prove its correctness.
3. Prove that the worst-case run time for HEAP-SORT is $\Theta(n \log n)$.
4. If we were to employ divide and conquer approach to design a comparison sort, demonstrate various sorting algorithms when the pivot element q is always chosen to be
   (a) mid-point of the array in each , i.e. MERGE-SORT,
   (b) such that $A[1 : q-1] \leq A[q] \leq A[q+1 : n]$, i.e. QUICK-SORT,

   for a given input array $A = [1,\ 4,\ 7,\ 6,\ 3,\ 9,\ 5,\ 2,\ 4,\ 6]$.

Section 5 Sorting in Linear Time

1. Prove that the running time for comparison sorts is $\Omega(n \log n)$.
2. Demonstrate COUNTING-SORT for the input array $A = [1,\ 4,\ 2,\ 5,\ 3,\ 1,\ 2,\ 4,\ 3,\ 1,\ 4]$.

Section 6 Dynamic Programming

1. Consider a 0-1 Knapsack problem with n indivisible items, where v_i and w_i are the value and weight of the i^{th} item respectively. If the size of the Knapsack is W, show that the Bellman equation is

$$V[i,j] = \min_{x_i \in \{0,1\}} v_i x_i + V[i-1, j - x_i w_i], \text{ for all } i = 1, \cdots, n \text{ and } j = 0, 1, \cdots, W,$$

   and write the pseudocode for the dynamic programming solution to 0-1 Knapsack problem.

2. Consider a matrix chain multiplication $A_1 \cdot A_2 \cdots A_n$, where A_i is a $p_{i-1} \times p_i$ matrix. IF the cost of computing the product of $k \times \ell$ and $\ell \times m$ matrices is $k\ell m$, show that the Bellman equation is

$$m[i,j] = \begin{cases} m[i,k] + m[k+1,j] + p_{i-1}p_k p_j, & \text{if } i > j \\ 0, & \text{otherwise.} \end{cases}$$

   Using the above Bellman equation, write a pseudocode for the dynamic programming algorithm for matrix-chain multiplication problem.

Section 7 Greedy Algorithms

1. When can a greedy algorithm produce an optimal solution?

2. Demonstrate on the following example, how to find an optimal solution to the Fractional Knapsack problem using a greedy algorithm?

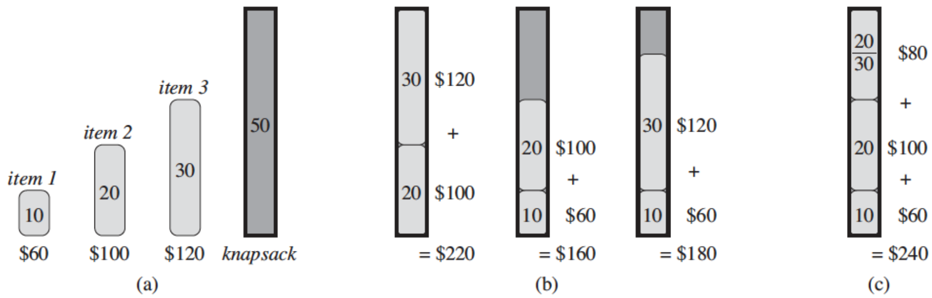

Figure 1: Fractional Knapsack

Section 8 Graph Search/Traversal

1. Demonstrate Breadth-first search (BFS) on a graph, such as the one shown in Figure 2.

2. Demonstrate Depth-first search (DFS) on a graph, such as the one shown in Figure 2.

3. What is the main difference between the implementation of BFS and DFS algorithms?

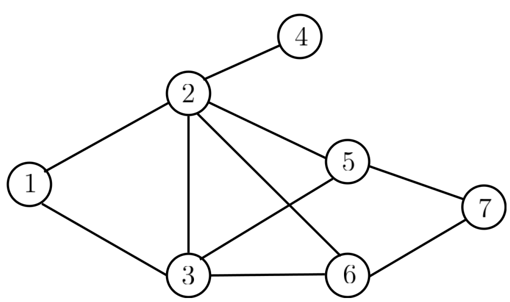

Figure 2: Graph Search

Section 9 Minimum Spanning Trees

1. Demonstrate Kruskal's algorithm on the graph, such as the one shown in Figure 3.

2. Demonstrate Prim's algorithm on the graph, such as the one shown in Figure 3.

3. Explain why dynamic programming is not a good approach to find minimum spanning trees?

4. State the primary difference between Kruskal's and Prim's algorithms?

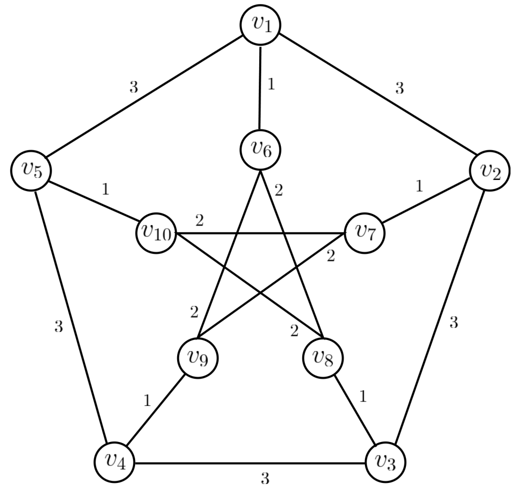

Figure 3: Minimum Spanning Trees

Section 10 Single-Source Shortest-Path Algorithms

1. Demonstrate Bellman-Ford algorithm on the graph, such as the one shown in Figure 4.
2. Demonstrate Dijkstra's algorithm on the graph, such as the one shown in Figure 4.
3. Explain how Bellman-Ford algorithm can be used to identify negative-weight cycless?

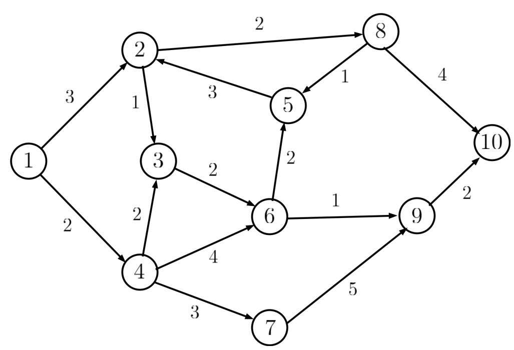

Figure 4: Directed Graph

Section 11 Max-Flow

1. Consider the directed graph in Figure 4. Assuming that the weights shown are edge capacities, demonstrate Ford-Fulkerson algorithm. Elaborate each step along with the residual graph, and identify the augmented flow path.

Section 12 NP Completeness

1. What is P, NP, NP-Hard, NP-Complete, and R classes?

2. Give one example problem in P, EXP, NP-Complete and R class.
3. What sequence of reductions are needed to prove that traveling salesman problem belongs to NP complete class?