Problem 1: True or False [20 points, 4 points per question] For each of the **claims** below, answer True or False. **NO justification is needed.**

Q1. **Claim:** $n^{2.1} = O(n^2 \log n)$.

False

Q2. **Claim:** For the stable matching problem, there can be more than 1 solution

True

Q3. **Claim:** Any graph G with no cycles has exactly $n - 1$ edges.

False

(This sample exam was from previous semester. In that semester, it is not assumed that graphs are connected graphs. So, the answer was false for that semester.

But in this semester, we assume all the graphs are connected graphs throughout the semester, so the answer for this question should be true in this semester.)

Q4. **Claim:** If all edges in a graph have weight 1, then there is an $O(m + n)$ time algorithm to find the minimum spanning tree, where m is the number of edges and n is the number of vertices.

True

Q5. **Claim:** Let T be a breadth-first search tree of an undirected graph G. Let (x, y) be an edge of G that is not an edge of T, then one of x or y is an ancestor of the other.

False

Continue on the next page ...

Problem 2: Short Answer [20 points, 5 points per question] Answer each question below. **NO justification is needed unless the problem ask to explain the reason.**

Q6. Give solution to the following recurrence.

$$T(n) = 6T(n/3) + n^5$$

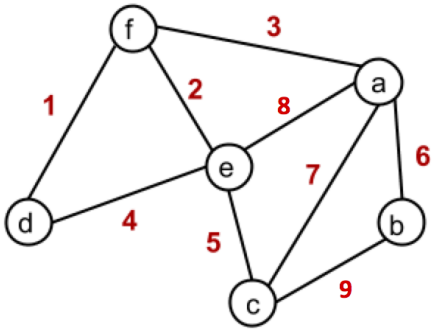
Answer: $T(n) = O(n^5)$

Q7. Give solution to the following recurrence.

$$T(n) = T(n/3) + 100n^{10}$$

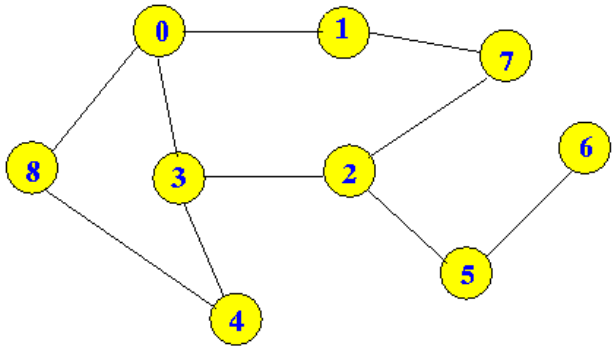
Answer: $T(n) = O(n^{10})$

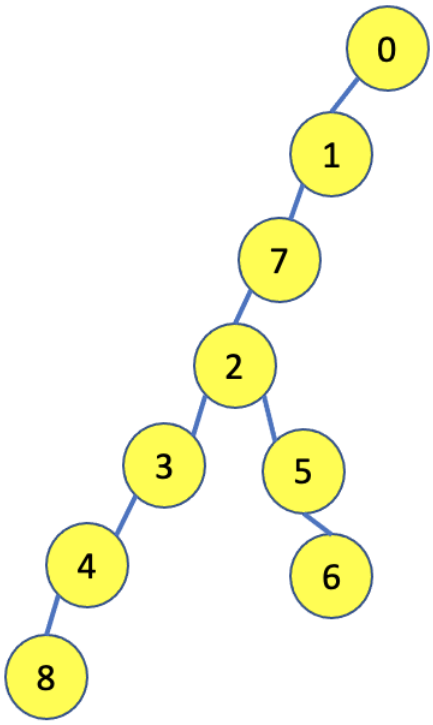
Q8. What is the total weight (sum of edge weights) for a minimum spanning tree of the graph below (red numbers represent edge weights)?



Answer: 17

Q9. Write a DFS tree for the following graph with vertex 0 as the start vertex.




0
1
7
2
3
5
4
6
8

.

Problem 3: Graph Visit [22 points]
Q10. Given a directed graph $G = (V, E)$ with no cycle. Give an time algorithm to check if there is a directed path that touches every vertex exactly once.
Hints: You can use the fact that topological sort can be done in $O(|E| + |V|)$ time.
Solution: Algorithm:

1. Run topological sort on G. Let $v_1, v_2, \ldots, v_{|V|}$ denote the resulted vertex sequence.
2. Output Yes if there is an edge from v_i to v_{i+1} for every $1 \leq i \leq |V| - 1$, otherwise output No.

Running time: $O(|V| + |E|)$.

Continue on the next page ...

Problem 4: Randall's Pokemon [19 points]
Randall has n pokemons and n poke balls. The power of i-th pokemon is k_i, and the power j-th poke ball is p_j. If Randall uses j-th poke ball to catch i-th pokemon, the reward is $k_i \cdot p_j$. Now, Randall wants to use the poke balls to catch the pokemons such that every poke ball can only be used once, and every pokemon can only be caught once. Randall wants to maximize the total reward, i.e., the sum of the reward by catching each pokemon using a poke ball.
Assume there are two pokemons with $k_1 = 2$ and $k_2 = 4$, and two poke balls with $p_1 = 3$ and $p_2 = 1$. What is the max total reward Randall can obtain?
Assume there are three pokemons with $k_1 = 2$, $k_2 = 7$, $k_3 = 5$, and three poke balls with $p_1 = 5$, $p_2 = 1$ and $p_3 = 10$. What is the max total reward Randall can obtain?
Example: Assume there are two pokemons with $k_1 = 2$ and $k_2 = 4$, and two poke balls with $p_1 = 3$ and $p_2 = 1$. The max total reword is obtained by using the first poke ball to catch the second pokemon, and the second poke ball to catch the first pokemon, which is $3 \cdot 4 + 1 \cdot 2 = 14$.

Q11. [15 points] Describe an algorithm to compute the max total reward Randall can obtain. No justification of running time and correctness is required here.

(a) Sort all the pokemons such that $k_1 \geq k_2 \geq \cdots \geq k_n$. Sort all the poke balls such that $p_1 \geq p_2 \geq \cdots \geq p_n$.

(b) return $\sum_{i=1}^{n} k_i \cdot p_i$.

Q12. [4 points] What is the running time of your algorithm? No justification is needed.

Answer: $O(n \log n)$.

Continue on the next page ...

Problem 5: Max Contiguous Subarray [19 points]
You are given a one dimensional array that may contain both positive and negative integers. Give an $O(n \log n)$ time algorithm to find the sum of contiguous (i.e. next to one another, in sequence) subarray of numbers which has the largest sum. You will receive partial score for slower algorithms.
For example, if the given array is [-2, -5, 6, -2, -3, 1, 5, -6], then $[6, 1, 5]$ is not a contiguous subarray, because there are other elements between 6 and 1. $[-5, 6, -2, -3]$ is a contiguous subarray, but its sum is not maximized. The maximum contiguous subarray sum is 7 (with respect to subarray $[6, -2, -3, 1, 5]$).
Solution: **Algorithm:**
Given an input array A, we use $A[i, j]$ to denote the contiguous subarray from i-th element to j-th element. Assume $A[1]$ is the first element and $A[n]$ is the last element for an array with n elements.
Routine SubarraySum takes an array A as input, and output m, s, t, sum defined as follows:

- m: maximum contiguous subarray sum of A
- s: the maximum contiguous subarray sum for all the subarrays containing the first element of A. If sums of all the subarrays containing the first element of A are negative, then s is 0.
- t: the maximum contiguous subarray sum for all the subarrays containing the last element of A. If sums of all the subarrays containing the last element of A are negative, then t is 0.
- sum: sum of all the elements of A

SubarraySum(array A)

1. $n \leftarrow$ length of A
2. If $n = 1$ and $A[1] \geq 0$, return $(A[1], A[1], A[1], A[1])$
3. If $n = 1$ and $A[1] < 0$, return (0, 0, 0, A[1])
4. Let $(m_1, s_1, t_1, sum_1) \leftarrow SubarraySum(A[1, \lfloor n/2 \rfloor])$,
   $(m_2, s_2, t_2, sum_2) \leftarrow SubarraySum(A[\lfloor n/2 \rfloor + 1, n])$
5. Return $(\max\{m_1, m_2, t_1 + s_2\}, \max\{s_1, sum_1 + s_2\}, \max\{t_2, sum_2 + t_1\}, sum_1 + sum_2)$

Running time:
$O(n)$ by Master Theorem.