CS 231: Fundamental Algorithms **Fall 2021**

Assignment #5

Your name:

Collaborators:

Total time spent:

Problem 1. [5 points]
Consider an algorithm whose running time $T(n)$ on an input of size n satisfies the following recurrence: $T(n) = aT(\frac{n}{b}) + cn$ where we assume the recurrence holds when $n > 1$, and that $T(1) = c$.

(a) How many nodes are there at level i of the recursion tree?
(b) What is the input size for a problem at level i of the recursion tree?
(c) How much work is done in a single function call at level i of the recursion tree? (Just as in class, count only the work done in the function itself, excluding recursive calls.)
(d) What is the total work done at level i of the recursion tree?
(e) How many levels are in the recursion tree?

Problem 2. [10 points]
Recall the problem of finding the number of inversions. As in the text, we are given a sequence of n numbers $a_1, ..., a_n$, which we assume are all distinct, and we define an inversion to be a pair $i < j$ such that $a_i > a_j$. We motivated the problem of counting inversions as a good measure of how different two orderings are. However, one might feel that this measure is too sensitive. Let's call a pair a significant inversion if $i < j$ and $a_i > 2a_j$. Write an $O(nlogn)$ algorithm to count the number of significant inversions between two orderings.

Problem 3. [10 points]
Let $G = (V, E)$ be an undirected graph with n nodes. A subset of the nodes is called an independent set if no two of them are joined by an edge. Finding maximal independent sets is difficult in general; but here we'll see that it can be done efficiently if the graph is "simple" enough.

Call a graph $G = (V, E)$ a path if its nodes can be written as $v_1, v_2, ..., v_n$, with an edge between v_i and v_j if and only if the numbers i and j differ by exactly 1. With each node v_i, we associate a positive integer weight w_i.

Consider, for example, the five-node path drawn in Figure 6.28. The weights are the numbers drawn inside the nodes. The goal in this question is to solve the following problem: Find an independent set in a path G whose total weight is as large as possible.

(a) Give an example to show that the following algorithm does not always find an independent set of maximum total weight.

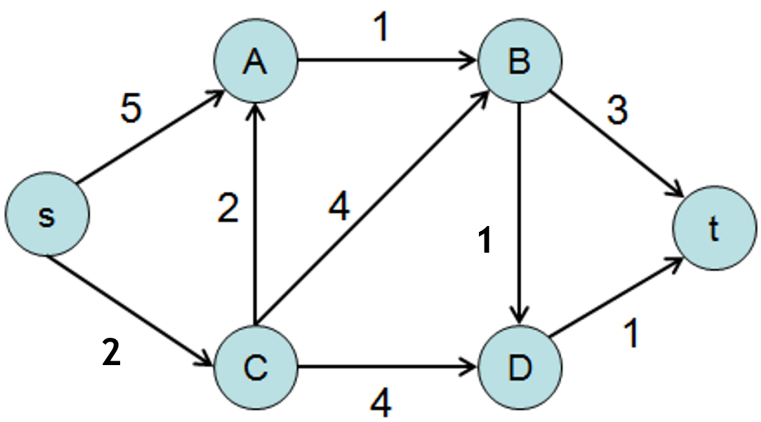
```
The "heaviest-first" greedy algorithm
  Start with S equal to the empty set
  While some node remains in G
    Pick a node v_i of maximum weight
    Add v_i to S
    Delete v_i and its neighbors from G
  Endwhile
  Return S
```

(b) Give an example to show that the following algorithm also does not always find an independent set of maximum total weight.

```
Let S_1 be the set of all v_i where i is an odd number
Let S_2 be the set of all v_i where i is an even number
(Note that S_1 and S_2 are both independent sets)
Determine which of S_1 or S_2 has greater total weight,
  and return this one
```

(c) Give an algorithm that takes an n-node path G with weights and returns an independent set of maximum total weight. **Analyze** running time complexity of your algorithm.

Problem 4. [8 points]
Given the code for Dijkstra's algorithm shared with you in class, and in your textbook, find the cost of the shortest paths from node s to the rest of the nodes in the graph shown below. Make sure to trace the algorithm exactly as shown in the algorithm. To help you trace it, I have provided you with this table to complete with every iteration.



Hint: An initial dummy iteration is already done for your to get you started, with all temporary distances initially set to ∞.

S	$V - S$	$d'(v), \forall v \in V - S$	Node to be added to S
$\{\}$	$\{s, A, B, C, D, t\}$	$d'(s) = 0, d'(A) = d'(B) = d'(C) = d'(D) = d'(t) = \infty$	s
$\{s\}$			

Finally, fill out the distances table below.

Node	Shortest Distance from s
A	
B	
C	
D	
t	

Problem 5. [10 points]
You are in an escape room tournament, which has n rooms numbered $1, 2, 3, ..., n$. Escaping from a room i is worth r_i points. According to the rules of the tournament, you must go through the rooms in order, but you can choose to skip some rooms. The reason you might choose to do this is that even though you can go through each room and escape to get the r_i points, some rooms have a penalty of not allowing you to go through any of the following f_i rooms. So you need to plan carefully.

Suppose that you are given the r_i and f_i values for all the rooms as input before the start of the tournament. **Write** the most efficient algorithm you can for choosing the set of rooms that maximizes your total points, and **compute** its asymptotic worst-case running time as a function of n.

Problem 6. [7 points]
Let us say that a graph $G = (V, E)$ is a near-tree if it is connected and has at most $n + 8$ edges, where $n = |V|$. a) **Write an algorithm** with running time $O(n)$ that takes a near-tree G with costs on its edges, and returns a minimum spanning tree of G. You may assume that all the edge costs are distinct.
b) **Explain** why the running time of your algorithm is $O(n)$.