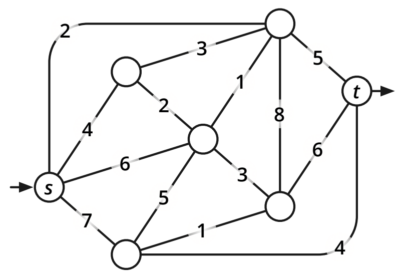
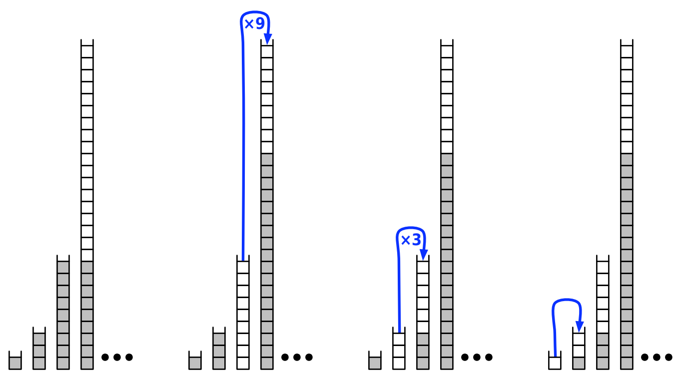
This exam lasts 180 minutes.
Write your answers in the separate answer booklet.
Please return this question sheet and your cheat sheets with your answers.

1. Clearly indicate the following structures in the weighted graph pictured below. Some of these subproblems have more than one correct answer.

   (a) A depth-first spanning tree rooted at s
   (b) A breadth-first spanning tree rooted at s
   (c) A shortest-path tree rooted at s
   (d) A minimum spanning tree
   (e) A minimum (s, t)-cut



2. A *multistack* consists of an infinite series of stacks $S_0, S_1, S_2, \ldots$, where the ith stack S_i can hold up to 3^i elements. Whenever a user attempts to push an element onto any full stack S_i, we first pop all the elements off S_i and push them onto stack S_{i+1} to make room. (Thus, if S_{i+1} is already full, we first recursively move all its members to S_{i+2}.) Moving a single element from one stack to the next takes $O(1)$ time.



Making room for one new element in a multistack.

   (a) In the worst case, how long does it take to push one more element onto a multistack containing n elements?
   (b) ***Prove*** that the amortized cost of a push operation is $O(\log n)$, where n is the maximum number of elements in the multistack.

3. Suppose we are given an array $A[1..n]$ of numbers with the special property that $A[1] \geq A[2]$ and $A[n-1] \leq A[n]$. A ***local minimum*** is an element $A[i]$ such that $A[i-1] \geq A[i]$ and $A[i] \leq A[i+1]$. For example, there are six local minima in the following array:

9	*7*	7	2	*1*	3	7	5	*4*	7	*3*	*3*	4	8	*6*	9

Describe and analyze an algorithm that finds a local minimum in the array A in $O(\log n)$ time.

4. Suppose we are given an n-digit integer X. Repeatedly remove one digit from either end of X (your choice) until no digits are left. The *square-depth* of X is the maximum number of perfect squares that you can see during this process. For example, the number 32492 has square-depth 3, by the following sequence of removals:

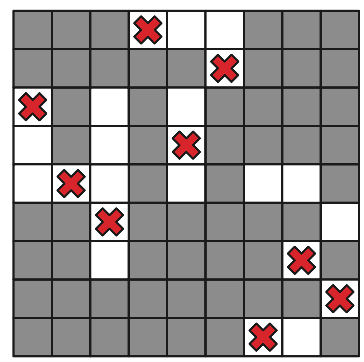
$$32492 \to \underline{\mathbf{3249}}\not{2} \to \underline{\mathbf{324}}\not{9} \to \not{3}24 \to \not{2}\underline{\mathbf{4}} \to \not{4}.$$

   Describe and analyze an algorithm to compute the square-depth of a given integer X, represented as an array $X[1\,..\,n]$ of n decimal digits. Assume you have access to a subroutine IsSquare that determines whether a given k-digit number (represented by an array of digits) is a perfect square ***in $O(k^2)$ time***.

5. Suppose we are given an $n \times n$ square grid, some of whose squares are colored black and the rest white. Describe and analyze an algorithm to determine whether tokens can be placed on the grid so that

   - every token is on a white square;
   - every row of the grid contains exactly one token; and
   - every column of the grid contains exactly one token.

   Your input is a two dimensional array $IsWhite[1\,..\,n, 1\,..\,n]$ of booleans, indicating which squares are white. (You solved an instance of this problem in the last quiz.)



6. Recall the following problem from Homework 2:

   - 3WayPartition: Given a set X of positive integers, determine whether there are three disjoint subsets $A, B, C \subseteq X$ such that $A \cup B \cup C = X$ and

$$\sum_{a \in A} a = \sum_{b \in B} b = \sum_{c \in C} c.$$

   (a) ***Prove*** that 3WayPartition is NP-hard. *[Hint: Don't try to reduce from 3SAT or 3COLOR; in this rare instance, the 3 is just a coincidence.]*

   (b) In Homework 2, you described an algorithm to solve 3WayPartition in $O(nS^2)$ time, where S is the sum of all elements of X. Why doesn't this algorithm imply that P=NP?

7. Describe and analyze efficient algorithms to solve the following problems:

   (a) Given an array of n integers, does it contain two elements a, b such that $a + b = 0$?

   (b) Given an array of n integers, does it contain three elements a, b, c such that $a + b + c = 0$?