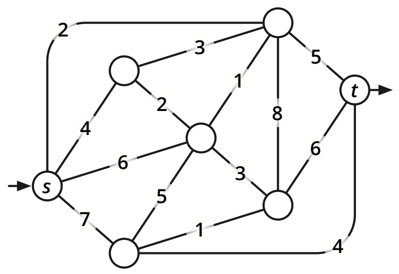
This exam lasts 180 minutes.
Write your answers in the separate answer booklet.
Please return this question sheet and your cheat sheets with your answers.

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. Describe a data structure that stores a set of numbers (which is initially empty) and supports the following operations in $O(1)$ amortized time:

   - INSERT(x): Insert x into the set. (You can safely assume that x is not already in the set.)
   - FINDMIN: Return the smallest element of the set (or NULL if the set is empty).
   - DELETEBOTTOMHALF: Remove the smallest $\lceil n/2 \rceil$ elements the set. (That's smallest by *value*, not smallest by *insertion time*.)

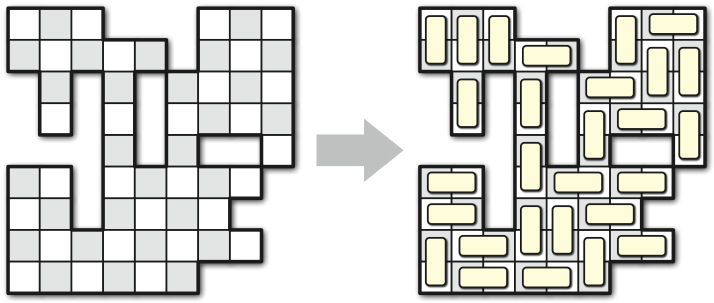
3. Suppose we are given two *sorted* arrays $A[1..n]$ and $B[1..n]$ containing $2n$ distinct numbers. Describe and analyze an algorithm that finds the nth smallest element in the union $A \cup B$ in $O(\log n)$ time.

4. Suppose you have a black-box subroutine QUALITY that can compute the 'quality' of any given string $A[1..k]$ in $O(k)$ time. For example, the quality of a string might be 1 if the string is a Québecois curse word, and 0 otherwise.

   Given an arbitrary input string $T[1..n]$, we would like to break it into contiguous substrings, such that the total quality of all the substrings is as large as possible. For example, the string `SAINTCIBOIREDESACRAMENTDECRISSE` can be decomposed into the substrings `SAINT + CIBOIRE + DE + SACRAMENT + DE + CRISSE`, of which three (or possibly four) are *sacres*.

   Describe an algorithm that breaks a string into substrings of maximum total quality, using the QUALITY subroutine.

5. Suppose you are given an $n \times n$ checkerboard with some of the squares removed. You have a large set of dominos, just the right size to cover two squares of the checkerboard. Describe and analyze an algorithm to determine whether one can place dominos on the board so that each domino covers exactly two squares (meaning squares that have *not* been removed) and each square is covered by exactly one domino. Your input is a two-dimensional array $Removed[1..n, 1..n]$ of booleans, where $Removed[i, j] = \text{TRUE}$ if and only if the square in row i and column j has been removed. For example, for the board shown below, your algorithm should return TRUE.



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 $a + b + c = 0$?