You have 120 minutes to answer four of these questions.
Write your answers in the separate answer booklet.
You may take the question sheet with you when you leave.

1. Consider the following algorithm for finding the smallest element in an unsorted array:

```
RandomMin(A[1..n]):
  min ← ∞
  for i ← 1 to n in random order
      if A[i] < min
          min ← A[i]    (⋆)
  return min
```

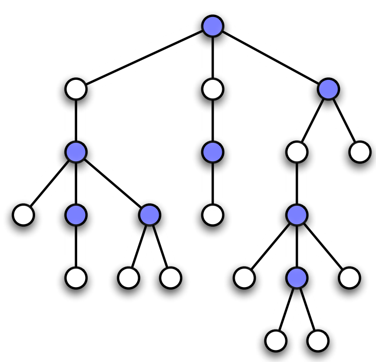
   (a) **[1 pt]** In the worst case, how many times does RandomMin execute line $(\star)$?

   (b) **[3 pts]** What is the probability that line $(\star)$ is executed during the last iteration of the for loop?

   (c) **[6 pts]** What is the *exact* expected number of executions of line $(\star)$? (A correct $\Theta(\,)$ bound is worth 4 points.)

2. Describe and analyze an efficient algorithm to find the size of the smallest vertex cover of a given tree. That is, given a tree T, your algorithm should find the size of the smallest subset C of the vertices, such that every edge in T has at least one endpoint in C.

   The following hint may be helpful. Suppose C is a vertex cover that contains a leaf ℓ. If we remove ℓ from the cover and insert its parent, we get another vertex cover of the same size as C. Thus, there is a minimum vertex cover that includes none of the leaves of T (except when the tree has only one or two vertices).



A tree whose smallest vertex cover has size 8.

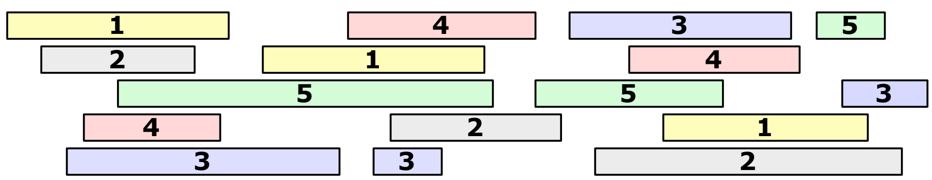
3. A *dominating set* for a graph G is a subset D of the vertices, such that every vertex in G is either in D or has a neighbor in D. The MINDOMINATINGSET problem asks for the size of the smallest dominating set for a given graph.

   Recall the MINSETCOVER problem from lecture. The input consists of a *ground set* X and a collection of subsets $S_1, S_2, \ldots, S_k \subseteq X$. The problem is to find the minimum number of subsets S_i that completely cover X. This problem is NP-hard, because it is a generalization of the vertex cover problem.

   (a) **[7 pts]** Describe a polynomial-time reduction from MINDOMINATINGSET to MINSETCOVER.

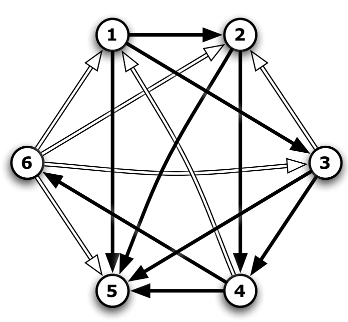
   (b) **[3 pts]** Describe a polynomial-time $O(\log n)$-approximation algorithm for MINDOMINATINGSET. *[Hint: There is a two-line solution.]*

4. Let X be a set of n intervals on the real line. A *proper coloring* of X assigns a color to each interval, so that any two overlapping intervals are assigned different colors. Describe and analyze an efficient algorithm to compute the minimum number of colors needed to properly color X. Assume that your input consists of two arrays $L[1..n]$ and $R[1..n]$, where $L[i]$ and $R[i]$ are the left and right endpoints of the ith interval. As usual, if you use a greedy algorithm, you must prove that it is correct.



A proper coloring of a set of intervals using five colors.

5. The *linear arrangement problem* asks, given an n-vertex directed graph as input, for an ordering $v_1, v_2, \ldots, v_n$ of the vertices that maximizes the number of *forward edges*: directed edges $v_i \to v_j$ such that $i < j$. Describe and analyze an efficient 2-approximation algorithm for this problem.



A directed graph with six vertices with nine forward edges (black) and six backward edges (white)