1. Suppose we are given a set of n rectangular boxes, each specified by their height, width, and depth in centimeters. All three dimensions of each box lie strictly between 10cm and 20cm, and all $3n$ dimensions are distinct. As you might expect, one box can be nested inside another if the first box can be rotated so that is is smaller in every dimension than the second box. Boxes can be nested recursively, but two boxes cannot be nested side-by-side inside a third box. A box is *visible* if it is not nested inside another box.

   Describe and analyze an algorithm to nest the boxes, so that the number of visible boxes is as small as possible.

2. Consider the following randomized version of mergesort. The input is an unsorted array $A[1..n]$ of distinct numbers. The MERGE subroutine takes two sorted arrays as input and returns a single sorted array, containing the elements of both input arrays, in linear time.

```
RandomizedMergeSort(A[1..n]):
  if n ≤ 1
      return A
  ℓ ← 0;  r ← 0
  for i ← 1 to n
      with probability 1/2
          ℓ ← ℓ + 1
          L[ℓ] ← A[i]
      else
          r ← r + 1
          R[r] ← A[i]
  L ← RandomizedMergeSort(L[1..ℓ])
  R ← RandomizedMergeSort(R[1..r])
  return Merge(L, R)
```

   (a) What is the probability that two distinct items $A[i]$ and $A[j]$ are both copied into L?

   (b) What is the probability that two distinct items $A[i]$ and $A[j]$ appear in the same subproblem for more than k levels of recursion?

   (c) What is the expected number of pairs of items that appear in the same subproblem for more than k levels of recursion?

   (d) Upper-bound the probability that ***at least one*** pair of items appear in the same subproblem for more than k levels of recursion. Equivalently, upper bound the probability that the recursion tree of RANDOMIZEDMERGESORT has depth at most k.

   (e) For what value of k is the probability in part (d) at most $1/n$?

   (f) Prove that RANDOMIZEDMERGESORT runs in $O(n \log n)$ time with high probability.

3. Suppose we are given a set R of n *red* points, a set G of n *green* points, and a set B of n *blue* points; each point is given as a pair (x, y) of real numbers. We call these sets *separable* if there is a pair of parallel lines $y = ax + b$ and $y = ax + b'$ such that (1) all red points are below both lines, (2) all blue points are above both lines, and (3) all green points are between the lines.

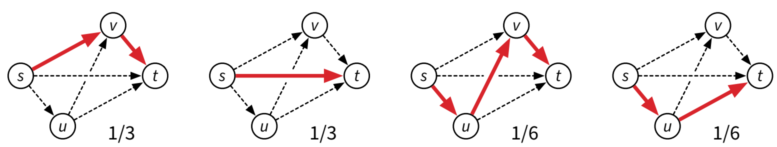
   (a) Describe a linear program that is feasible if and only if the point sets G, B, R are separable.

   (b) Describe a linear program whose solution describes a pair of parallel lines that separates G, B, R whose *vertical* distance is as small as possible. (Here you can assume that G, B, R are separable.)

   [Hint: Don't try to solve these problems; just describe the linear programs.]

4. Let $G = (V, E)$ be an arbitrary dag with a unique source s and a unique sink t. Suppose we compute a random walk from s to t, where at each node v (except t), we choose an outgoing edge $v \rightarrow w$ uniformly at random to determine the successor of v.

   For example, in the following four-node graph, there are four walks from the unique source s to the unique sink t, chosen with the indicated probabilities:



   (a) Describe and analyze an algorithm to compute, for every vertex v, the probability that the random walk visits v. For example, in the graph shown above, a random walk visits the source s with probability 1, the bottom vertex u with probability $1/3$, the top vertex v with probability $1/2$, and the sink t with probability 1.

   (b) Describe and analyze an algorithm to compute the expected number of edges in the random walk. For example, given the graph shown above, your algorithm should return the number $1 \cdot 1/3 + 2 \cdot (1/2 + 1/6) + 3 \cdot 1/6 = 11/6$.

   Assume all relevant arithmetic operations can be performed exactly in $O(1)$ time.

5. You are managing a company with a large number of project teams. Each project team contains exactly three people. A single employee could belong to any number of project teams, but all of the project teams are different.

   You need to organize a meeting to make a major announcement. Because this announcement will affect every project team, you need to ensure that at least one member of every project team attends the meeting. On the other hand, there isn't enough space to invite every employee. Moreover, you expect the announcement to be unpopular, so you want to avoid inviting employees who are likely to cause trouble.

   For each employee x, you know a positive real number $\mathit{trouble}(x)$, which roughly estimates the probability that x will cause trouble. Your task is to choose a subset X of the employees that includes at least one member of each project team, such that $\sum_{x \in X} \mathit{trouble}(x)$ is as small as possible.

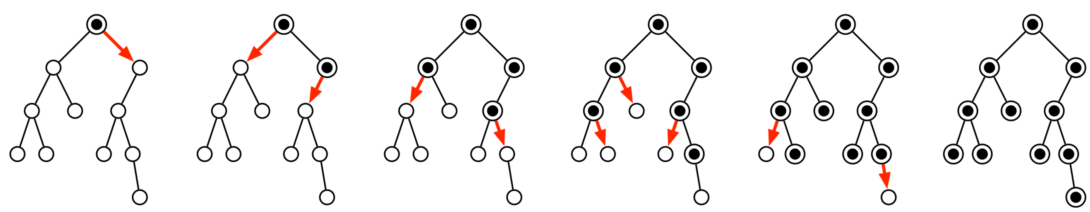
   (a) Write a linear programming relaxation for this problem.

   (b) Describe an efficient **3-*approximation*** algorithm for this problem. *[Hint: Round the solution to your linear program from part (a).]*

   A 3-approximation algorithm that is correct when $\mathit{trouble}(x) = 1$ for every employee x is worth half credit.

6. Suppose we need to distribute a message to all the nodes in a given binary tree. Initially, only the root node knows the message. In a single round, each node that knows the message is allowed (but not required) to forward it to at most one of its children. Describe and analyze an algorithm to compute the minimum number of rounds required for the message to be delivered to all nodes in the tree.

   For example, given the tree below as input, your algorithm should return the integer 5.



A message being distributed through a binary tree in five rounds.