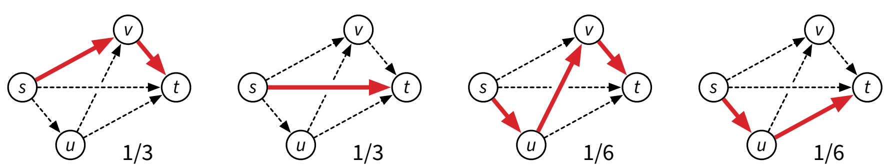
You have 180 minutes to answer six questions. There is no question 3.
Write your answers in the separate answer booklet.

0. During an unprecedented dos-equis-virus outbreak, the local emergency call center receives calls from n people who each need to be airlifted to one of k local hospitals. Each patient must be flown to a hospital within 20 miles of their home; however, we do not want to overload any single hospital.

   Describe an algorithm that assigns each patient to a hospital while keeping the maximum number of people flown to any single hospital as small as possible. Your input is an array $D[1..n, 1..k]$, where $D[i, j]$ is the distance in miles from patient i's home to hospital j. Analyze your algorithm as a function of n (the number of patients) and k (the number of hospitals).

1. 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 s to t, which are 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 $2 \cdot 1/3 + 1 \cdot 1/3 + 3 \cdot 1/6 + 2 \cdot 1/6 = 11/6$.

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

2. Consider the following randomized version of mergesort. The input is an unsorted array $A[1..n]$ of distinct numbers. Except in the base case, each element $A[i]$ is assigned to one of the two recursive subproblems according to a fair independent coin flip. 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) Fix two arbitrary indices $i \neq j$. What is the probability that $A[i]$ and $A[j]$ appear in the same recursive subproblem (either L or R)?

(b) What is the probability that $A[i]$ 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) Give an upper bound on 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 greater than 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 probability at least $1 - 1/n$.

4. Suppose we are given a *target* string $T[1..n]$ and an list of k *fragment* strings $F_1[1..m_1]$, $F_2[1..m_2], \dots, F_k[1..m_k]$. Describe and analyze an algorithm to find the shortest sequence of fragment strings F_i whose concatenation is the target string T. You can assume that such a sequence exists. The same fragment F_i can be used multiple times. Express the running time of your algorithm in terms of the parameters n, k, and $m = \sum_i m_i$.

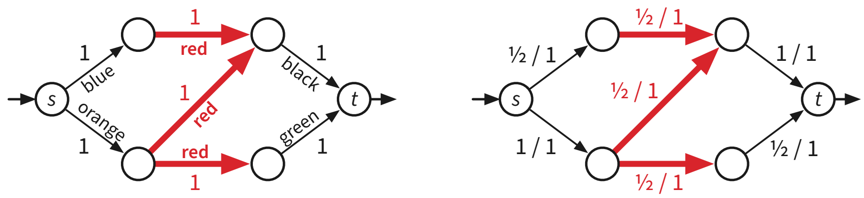
For example, suppose we are given the target string $T =$ ABRACADABRA and the fragment strings $F_1 =$ A, $F_2 =$ ABRA, $F_3 =$ ARC, $F_4 =$ BRAC, $F_5 =$ CAD, $F_6 =$ DAB, and $F_7 =$ RA. Then T can be decomposed into fragments in two different ways:

$$\text{ABRA} \cdot \text{CAD} \cdot \text{ABRA} = F_2 \cdot F_5 \cdot F_2.$$
$$\text{A} \cdot \text{BRAC} \cdot \text{A} \cdot \text{DAB} \cdot \text{RA} = F_1 \cdot F_4 \cdot F_1 \cdot F_6 \cdot F_7.$$

Your algorithm should return the integer 3, which is the length of the shorter decomposition.

5. Suppose we are given a standard flow network $G = (V, E)$, with a source vertex s, a target vertex t, and capacities $c(e) \geq 0$ for every edge e. Suppose each edge in G also has a *color*. A flow f in G is *color-consistent* if $f(e) = f(e')$ for every pair of edges e and e' that have the same color. The *maximum color-consistent flow* problem asks for a color-consistent flow with maximum value. The standard maximum flow problem is the special case where every edge has a different color.

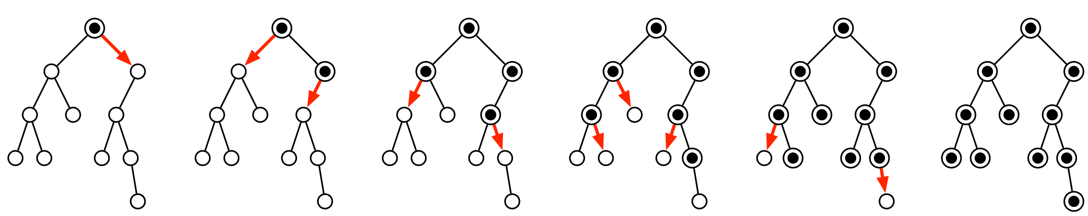
   As an example, consider the colored flow network shown below left. The three thick edges in the middle (forming a Z) all have the same color ("red"); the other four edges have distinct colors. Every edge has capacity 1. The unique maximum color-consistent flow in this network, shown below right, has value 1½.



   Describe a linear program whose solution is the maximum color-restricted flow in G. *[Hint: Modify the standard linear program for maximum flow. Don't try to actually compute this flow.]*

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.