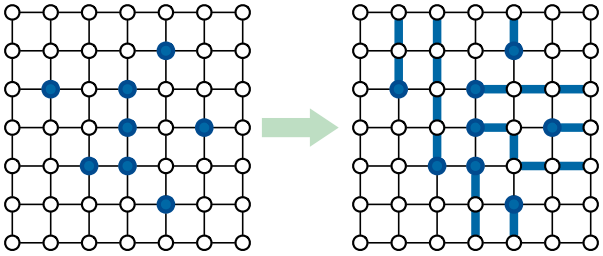
This is a reference version of an exam that was offered online via Gradescope.
There are some minor formatting differences from the actual exam.

---

1. An $n \times n$ grid is an undirected graph with n^2 vertices organized into n rows and n columns. Every vertex is connected to the nearest vertex (if any) above, below, to the right, and to the left.

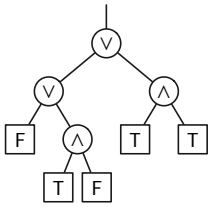
   Suppose m distinct vertices in the $n \times n$ grid are marked as *terminals*. The ***escape problem*** asks whether there are m vertex-disjoint paths in the grid that connect the terminals to any m distinct boundary vertices. Describe and analyze an efficient algorithm to solve the escape problem.

   For example, given the input on the left below, your algorithm should return TRUE.

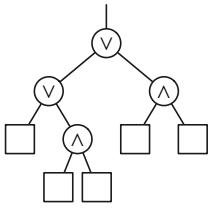


2. A *boolean expression tree* is a binary tree where every internal node has exactly two children, every internal node is labeled with one of the boolean operations $\wedge$ ("and") or $\vee$ ("or"), and every leaf is labeled either TRUE or FALSE.

   (a) Describe and analyze an algorithm to evaluate the boolean expression described by a given boolean expression tree. For example, given the following boolean expression tree, which describes the boolean expression $(F \vee (T \wedge F)) \vee (T \wedge T)$, your algorithm should return TRUE.



   (b) Now suppose you are are given a boolean expression tree without the leaf labels. Describe and analyze an algorithm to count the number of different ways to label the leaves so that the resulting expression evaluates to TRUE. For example, given the following labeled tree as input, your algorithm should return the integer 23.



3. A line of n passengers is waiting to board an airplane with n seats. Each passenger has an assigned seat. The very first passenger in line is an absent-minded tenured algorithms professor, who is distracted by planning his final exam and has forgotten his assigned seat. When the professor boards the plane, he chooses a seat uniformly at random and sits there.

   The remaining $n-1$ passengers then board the plane one at a time. When each passenger boards, if the professor is sitting in their assigned seat, the professor apologetically chooses a different *unoccupied* seat, again uniformly at random, and sits there. The next passenger enters only after the previous passenger and the professor are seated. After all n passengers have boarded, everyone is in their assigned seat, including the professor.

   (a) What is the exact probability that the professor never moves after choosing his first seat?

   (b) What is the exact probability that the kth passenger to board the plane has to ask the professor to move?

   (c) What is the exact expected number of times the professor changes seats?

   (d) What is the exact probability that the professor moves exactly once?

4. Describe and analyze an algorithm for string matching with *wildcards*. A wildcard is a special symbol `*` that can appear in the pattern but not in the text, and which matches any substring in the text. The input to your algorithm consists of the pattern array $P[1..m]$ and the text array $T[1..n]$.

   For example, given the pattern `ABR*CAD*BRA` and the text `SCHWABRAINCADBRANCH`, your algorithm should return TRUE, because the pattern matches the substring `ABRAINCADBRA`, woth the first wildcard matching `AIN` and the second wildcard matching the empty string.

5. Suppose you are given an array $A[1..n]$ of numbers, some of which are marked as *icky*. Describe and analyze an algorithm to compute the length of the longest increasing subsequence of A that includes at most k icky numbers. Your input consists of the array $A[1..n]$ of numbers, another boolean array $Icky[1..n]$, and the integer k.

   For example, suppose your input consists of the integer $k = 2$ and the following array (with icky numbers are indicated by stars):

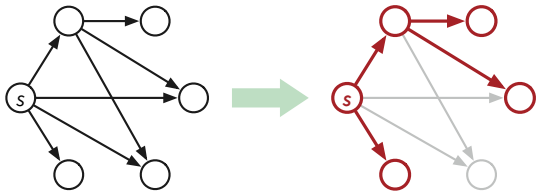
$3^\star$	$1^\star$	4	$1^\star$	$5^\star$	9	$2^\star$	6	5	$3^\star$	5	9	7	$9^\star$	3	2	3	$8^\star$	4	$6^\star$	2	$6^\star$

   Then your algorithm should return the integer 5, which is the length of the increasing subsequence $4, 5^\star, 6, 7, 9^\star$.

6. ***This problem is broken.*** Everybody who took the exam received full credit for this problem. In fact, this problem is NP-hard, but not trivially so; the first NP-hardness proof appears to have been published in late 2019.

   Suppose you are given a directed acyclic graph G with a single source vertex s. Describe an algorithm to find the size of the largest ***rooted binary subtree*** of G. Your algorithm is looking for a subgraph T of G with as many vertices as possible, such that every vertex in T except one (the root) has exactly one incoming edge in T, and every vertex in T has at most two outgoing edges in T. Your algorithm should return the number of vertices in T, not the actual subgraph.

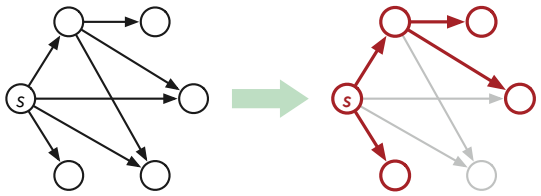
   For example, given the dag on the left below as input, your algorithm should return the integer 5, which is the size of the rooted binary tree on the right.



6̃. ***This is the question that we should have asked.***[6]

   Suppose you are given a directed acyclic graph G with a single source vertex s. Describe an algorithm to determine whether G contains a ***spanning binary tree***. Your algorithm is looking for a spanning tree T of G, such that every vertex in G has at most two outgoing edges in T and every vertex of G *except* s has exactly one incoming edge in T.

   For example, given the dag on the left below as input, your algorithm should FALSE, because the largest binary subtree excludes one of the vertices.



[6]It's pronounced "sñix".