Algorithms Spring 2020 Final

Once you download this final, you cannot communicate with anyone about this final or about algorithms until the deadline. Be aware that students can be given an extension under extenuating circumstances, so please be circumspect in what you share afterwards.

Solve 5 of the following 6 problems. Submit your solution in the D2L dropbox before Thursday 23:59 pm. The only acceptable format is a pdf.

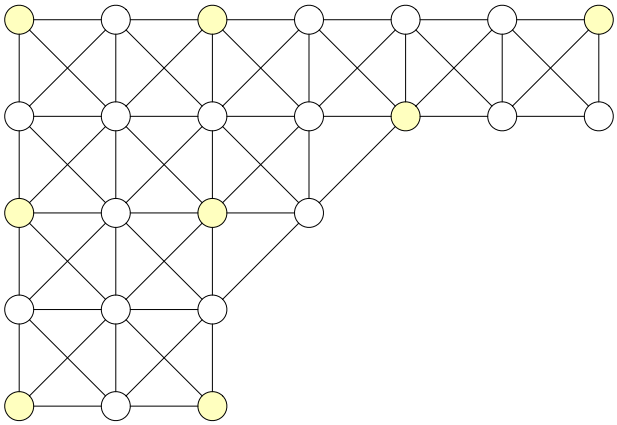
Problem 1:

For the following program, determine a recurrence for the number a_n of asterisks printed on input n. Then solve this recurrence.

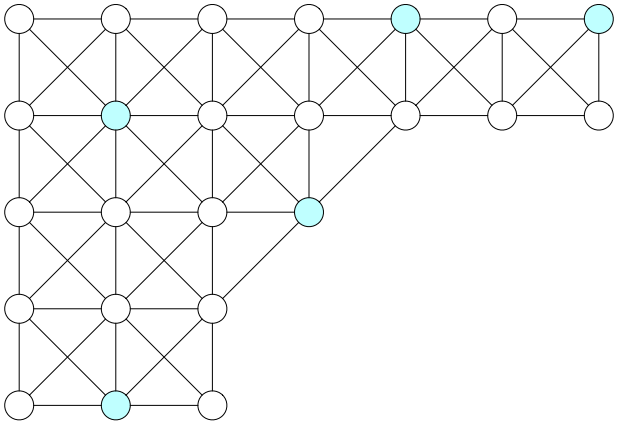
```
def aster(n):
   if n == 0:
      print('*', endl="")
      return
   for i in range(n*n):   #n-squared
      aster(n-1)
```

Problem 2:

A subset S of vertices in an undirected graph is called hegemonic, if any node is either in S or is the neighbor of a vertex in S. For example, in the following graph, the yellow vertices are a hegemonic set.



But with the same graph, we can have another hegemonic set (in blue). This shows that hegemonic sets are not only not uniquely determined but also can have different numbers of elements.

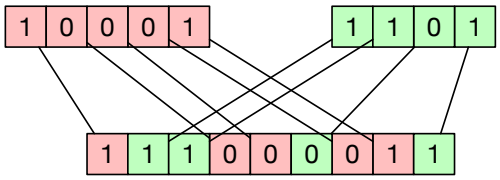


The problem of finding a hegemon of minimum size is NP-complete. **Show that the following *greedy algorithm* does not always find a hegemon of minimum size, but always finds a hegemon.**

def greedy_hegemon(G):
  If G is empty, return the empty set.
  Find the vertex v with most neighbors and put it into the hegemon set.
  Remove that vertex and all its neighbors from the set of vertices forming a subgraph G'.
  Return $\{v\} \cup$ greedy_hegemon(G')

Determine the run-time of greedy_hegemon(G) using Θ notation.

Problem 3:



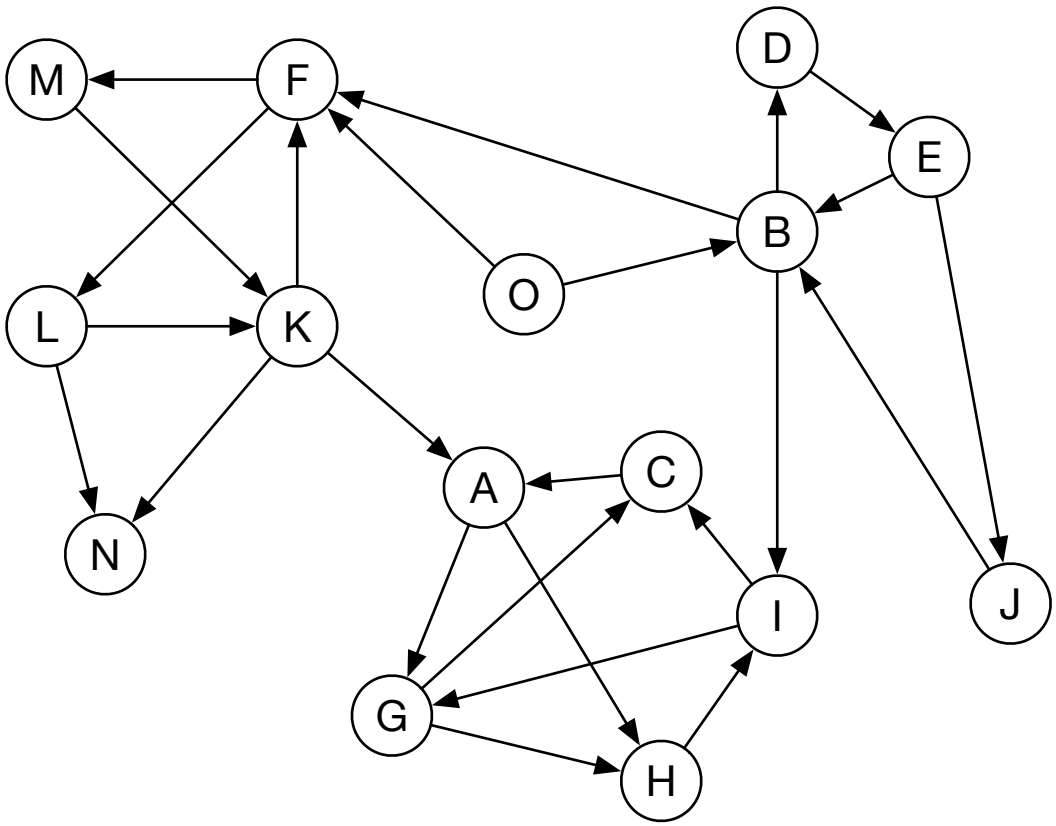
Two bit-strings $X = (x_1, x_2, x_3, \ldots x_n)$ and $Y = (y_1, y_2, y_3, \ldots y_m)$ produce an interleaving $Z = (z_1, z_2, \ldots z_{n+m})$ if the bits of Z are taken from X and Y, but appear in the same order as they appear there. For example, if $X = (1,0,0,0,1)$ and $Y = (1,1,0,1)$, then $Z = (111000011)$ is an interleaving (see below), but $Z' = (111100001)$ is not, even though it has the same letters. The latter is true because we can have most three leading ones in an interleaving before we need to get a zero.

Let $d(i, j)$ be a Boolean function that is true if $(z_1, z_2, \ldots z_{i+j})$ is an interleaving of $(x_1, x_2, \ldots, x_i)$ and $(y_1, y_2, \ldots, y_j)$. For example, $d(0,0) =$ False and $d(1,1)$ is True if and only if $z_1 = x_1 \wedge z_2 = y_1$ or $z_1 = y_1 \wedge z_2 = x_1$. Find a recursive formula that expresses $d(i, j)$ in terms of $d(i-1, j)$ and/or $d(i, j-1)$ and explain why this gives a dynamic programming solution to determining interleaving that works in time $O(nm)$.

Problem 4:

Execute the connected component algorithm on the following graph. Whenever you have a choice to make, use the alphabetic ordering of the nodes. E.g., when you select a node to visit, you first start with node A. When you visit A, you have two nodes that you can visit, so you choose G over H according to the alphabet.

(For your solution, show every step of the algorithm.)



Problem 5:

Find the 3-colorable graph representing the 3-SAT problem

$$(\neg a \vee b \vee c) \wedge (\neg a \vee \neg b \vee \neg c)$$

and color it with three colors such that no two neighboring nodes share the same color. Your graph should have three vertices for B, T, and F, three times two vertices for the pairs of literals, and two times 6 vertices for the clauses.

Problem 6:

Assume you are given an **ordered** array of n numbers in $\{0,1\}$. Develop an $O(\log n)$ algorithm to determine how many one-s there are in the array.