Name ____

CSCI 332, Fall 2025

Exam 2

Note that this exam has three sections. They are:

1. Greedy Algorithms (30 points)
2. Divide and Conquer (35 points)
3. Dynamic Programming (35 points)

You may use a double sided 3x5 handwritten notecard of notes during the test but no other resources. If you need more space than what is given, develop your solution on scratch paper before copying your final answer to the exam paper.

Good luck!

Section 1 (Greedy Algorithms)

1. You and your eight-year-old nephew Elmo decide to play a simple card game. At the beginning of the game, the cards are dealt face up in a long row. Each card is worth a different number of points, which could be positive, negative, or zero. After all the cards are dealt, you and Elmo take turns removing either the leftmost or rightmost card from the row, until all the cards are gone. At each turn, you can decide which of the two end cards to take. The winner of the game is the player that has collected the most points when the game ends.

   Having never taken an algorithms class, Elmo follows the obvious greedy strategy—when it's his turn, Elmo always takes the card with the higher point value. **Give an example set of cards where also playing a greedy strategy will cause you to lose to Elmo, when some other strategy would have allowed you to win.** Assume that Elmo goes first.

Recall the interval scheduling problem from class: we are given n jobs, each with a start time s and finish time f, and we want to select the maximum number of jobs that do not overlap.

2. Suppose there are $n = 9$ jobs with the following start and finish times: $s_A = 0, f_A = 6$; $s_B = 1, f_B = 4$; $s_C = 3, f_C = 5$; $s_D = 3, f_D = 8$; $s_E = 4, f_E = 7$; $s_F = 5, f_F = 9$; $s_G = 6, f_G = 10$; $s_H = 8, f_H = 11$; $s_I = 9, f_I = 12$.

   Give an example of a maximal set of non-overlapping jobs from this problem instance.

What follows is the inductive argument in a proof that the greedy strategy that selects the job with the earliest finish time that is compatible with previously selected jobs does always produce an optimal solution.

Proof: Let $G = (g_1 g_2, \dots, g_k)$ be a set of jobs chosen by the earliest-finish-time-first greedy strategy (sorted by start time) and let $X = (g_1, g_2, \dots, g_{j-1}, x_j, x_{j+1}, \dots, x_m)$ be an optimal set of jobs (sorted by start time) where x_j is the first job different from the jobs chosen in G.

We need to show that g_j can be swapped into G for x_j. We know that g_j has no conflicts with the previous jobs g_1 through g_{j-1}, since those are the jobs chosen by the greedy algorithm. But what about x_{j+1} through x_m? Since g_j has the earliest finish time of any job that does not conflict with jobs g_1 through g_{j-1}, g_j must finish earlier than x_j. So since x_j doesn't conflict with jobs x_{j+1} through x_m, it must be the case that g_j doesn't either. □

3. (4 points)

   Consider again the input from question 1. G for this example is the set of jobs (B, E, H). Let $X = (B, F, I)$, another optimal solution that is different from G.

   In this example, what are x_j and g_j?

   $x_j =$
   $g_j =$

   What are the finishing times of x_j and g_j?

   $f_{x_j} =$
   $f_{g_j} =$

Section 2 (Divide and Conquer)

In this section, we will think about the computational problem of where to insert a new element into a sorted array. Specifically, we will return the position *after which* to insert the element. For example, in the array $A = [5, 40, 44, 600]$, we would insert the element 41 after index 1. If we need to insert the new element before the current first element, we return -1.

4. Let $A = [-85, -45, -5, 6, 12, 18]$. After which index would we insert a new element 15? (You only get points if you give an index here, not "between x and y" or something similar.) Assume the indices start at 0.

5. Give a naive, $\Theta(n)$ worst-case time algorithm to solve the Insert problem. Remember to return the index after which to insert the new element.

```
NaiveInsert(sorted array $A$, new element $t$):
```

6. We can solve this problem using divide and conquer based on the following observation: given a sorted array A and a new element t, the index after which to insert t must either occur in (fill in the blanks):

   - ____, or
   - ____

The pseudocode to implement a divide-and-conquer Insert is shown below. Recall that $\lfloor x \rfloor$ takes the *floor* of x, meaning that it rounds down to the nearest integer.

```
Insert(sorted array $A$, new element $t$):
   min = 0
   max = length of $A$ - 1
   make $A$ and $t$ global
   return InsertRecursive(min, max)
```

```
InsertRecursive(index min, index max):
   If min = max:
         If $t < A$[min]:
               Return min -1
         Else:
               Return min
   Else:
         mid = $\lfloor$ (min + max) / 2 $\rfloor$
         If $t < A$[mid]:
               return InsertRecursive(min, mid - 1)
         Else:
               return InsertRecursive(mid + 1, max)
```

7. Suppose you have the length 16 array A below, with indices labeled above the array entries:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	3	5	6	12	100	105	108	133	229	277	280	551	559	601	603

and new element $t = 7$ and call Insert(A, t). Draw the recursion tree for the resulting call to InsertRecursive, labeling each node with the values of min and max for that call. The first node is given for you.

InsertRecursive(0, 15)

8. Describe a worst-case input for Insert, or explain why all inputs take asymptotically the same time.

9. What is the recurrence relation for the worst-case runtime of InsertRecursive on an array of length n?

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ \underline{\hspace{8cm}} & \text{if } n > 1. \end{cases}$$

10. What is the depth of the recursion tree for an input array of length n?

11. What is work done at each node?

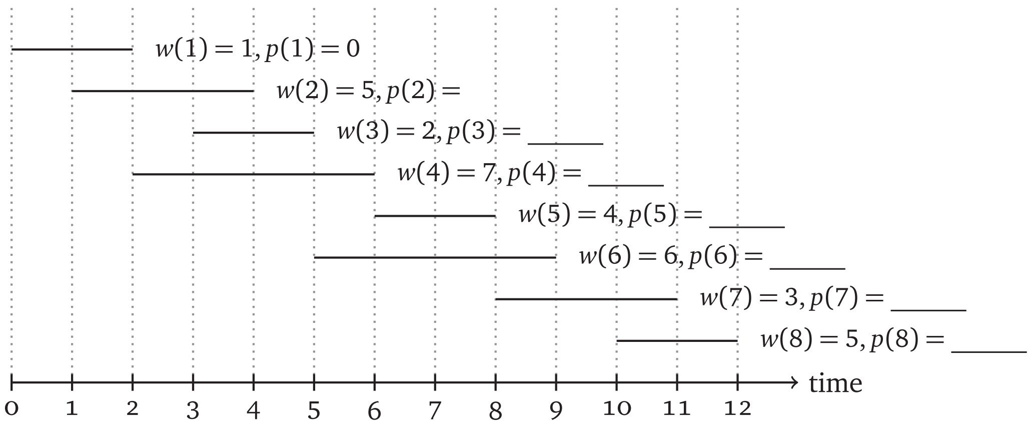
12. Give a function $f(n)$ such that the worst-case runtime of InsertRecursive is $\Theta(f(n))$.

Section 3 (Dynamic Programming)

Recall the *weighted interval scheduling problem* from class. In this problem, we are given n intervals, each with a start time $s(i)$, a finish time $f(i)$, and a weight $w(i)$. The goal is to find a set of mutually compatible intervals with maximum total weight.

We will also define $p(i)$ to be the index of the last interval that finishes before interval i starts (or 0 if there is no such interval).

13. For the example below, fill in the values of $p(3)$ through $p(8)$.



$w(1) = 1, p(1) = 0$
$w(2) = 5, p(2) =$
$w(3) = 2, p(3) =$ ______
$w(4) = 7, p(4) =$ ______
$w(5) = 4, p(5) =$ ______
$w(6) = 6, p(6) =$ ______
$w(7) = 3, p(7) =$ ______
$w(8) = 5, p(8) =$ ______
time
0 1 2 3 4 5 6 7 8 9 10 11 12

In order to give a dynamic programming solution to this problem, we follow the four steps:

14. English description of the subproblems:
Let $MaxWeight(i)$ be the ____
____.

15. You notice that the maximum weight of any set of compatible intervals at index i either:

- does not include interval i, so the maximum weight is just the maximum weight up to $i - 1$, or
- does include interval i, so the maximum weight is the weight of interval i plus the maximum weight up to $p(i)$.

Using this observation, fill in the recursive part of the recursive definition of the subproblem with the notation provided for w and p values.

$$MaxWeight(i) = \begin{cases} 0 & if\, i = 0 \\ \underline{\hspace{6cm}} & if\, i > 0 \end{cases}$$

16. To memoize this algorithm, we will use an array A of length ____________ (be careful! only exact answers will get full credit) to store the needed values of $MaxWeight(i)$ and fill it in increasing order.

17. Finally, we write the pseudocode for the algorithm using the above information. Use an array A to store the values of $MaxWeight(i)$. Assume that your inputs are sorted by finishing time already.

```
MaxIntervalWeight($s[1..n], f[1..n], w[1..n], p[1..n]$):
    # initialize an array A



    # do we have any base cases?





    # how should we fill in the rest of the array?








    # what should we return to get the final answer?
```

18. What are the values of *A* after running MaxIntervalWeight on the example above?