

Midterm Examination

CS 561 Data Structures and Algorithms
Fall, 2026

Name:
Email:

Directions:

- This exam lasts 75 minutes. You may use two regular-sized sheets of paper, handwritten on both sides. No other notes, books, or electronic devices are permitted.”
 - *Show your work!* You will not get full credit, if we cannot figure out how you arrived at your answer.
 - Write your solution in the space provided for the corresponding problem.
-

Question	Points	Score	Grader
1	20		
2	20		
3	20		
4	20		
5	20		
Total	100		

1. **Short Answer (4 points each)** Answer in *simplest possible Θ notation* unless otherwise specified.

(a) Each of n skip-list items reaches at least $3 \log n$ levels with probability n^{-3} . Using a union bound, what is an upper bound on the probability that any item does? (Answer in tightest big-O)

(b) Solution to the recurrence: $T(n) = 8T(n/2) + \sqrt{n}$

(c) Solution to the recurrence: $T(n) = 3T(n/2) + n^2$

(d) Solution to the recurrence: $f(n) = 5f(n-1) - 6f(n-2) + 2^n$
(answer in tightest big-O)

(e) Expected runtime of randomized quicksort on n distinct keys already in sorted order

2. **Probability and Expectation (20 points)** A purse contains 2 nickels and 10 dimes. The coins are drawn from the purse one at a time, uniformly at random without replacement, and you win all the dimes that are drawn between the 2 nickels.

Let X be a random variable giving the number of dimes drawn between the two nickels. For each dime, $i \in [1, 10]$, let X_i be an indicator random variable that is 1 if that dime is drawn between the nickels. Let N denote either nickel and d_i denote dime i . Then, for a fixed dime i , ignoring the other dimes, there are 3 possible arrangements, each of which is equally likely: N, N, d_i , N, d_i, N and d_i, N, N .

- (8 points) What is $E(X)$? Hint: use linearity of expectation.
- (2 points) Use Markov's inequality to bound the probability that you win at least 4 dimes.
- (10 points) The ten dimes are labeled $1, \dots, 10$. Instead of winning their face values, you receive i dollars if dime i is drawn between the nickels. Let Y be your total winnings. What is $E(Y)$?

3. **Induction (20 points)** A procedure starts with a group of $n \geq 1$ objects. When given a group of size 1, it does nothing at a cost of 0. When given a group of size $n \geq 2$, it splits the group into non-empty groups of size $n - a$ and a for some integer a with $1 \leq a \leq n - 1$, incurs a cost of $a(n - a)$, and then recurses on these 2 smaller groups. The choice of a may differ at every recursive call.

(a) (6 points) Write a recurrence relation for $T(n)$, the total cost incurred on a group of size n . Don't forget the base case.

(b) (14 points) Give an inductive proof showing that $T(n) = \binom{n}{2}$, which is $\frac{n(n-1)}{2}$, regardless of the choices of splits. Don't forget the BC, IH and IS. *Hint:* When objects are split into groups of sizes x and y , every unordered pair lies entirely in the first group, entirely in the second group, or has one object in each. Thus

$$\binom{x+y}{2} = \binom{x}{2} + \binom{y}{2} + xy.$$

Use this identity in your inductive step.

3. Induction (20 points), continued.

4. **Weighted Activity Selection (20 points)** In the weighted activity selection problem, you are given n activities, each having a start time, a finish time, and a positive weight. Two activities are compatible if one finishes at or before the other starts. Your goal is to compute the maximum total weight of a non-overlapping set of activities. Return only this optimal weight. For example, if the activities are given as a tuple $(starttime, endtime, weight)$, then we may have the input $(1, 3, 5), (1, 2, 3), (2, 3, 3)$, with optimal solution $(1, 2, 3), (2, 3, 3)$, which has total weight of 6.

(a) (4 points) Show that the greedy algorithm discussed in class where you always schedule the remaining activity with earliest finish time is not optimal for this weighted problem.

(b) (12 points) Consider the following approach to create a dynamic program. Sort all activities by nondecreasing finish time in order $a_1, a_2, \dots, a_n$. Let s_i , f_i , and w_i be the start time, finish time, and weight of activity a_i , respectively. For each $1 \leq i \leq n$, let $prior(i)$ be the largest index $j < i$ such that $f_j \leq s_i$, or 0 if no such index exists. Finally, for all $i, 1 \leq i \leq n$, let $W(i)$ be the maximum total weight achievable if we only consider activities $a_1, \dots, a_i$. Write a recurrence relation for $W(i)$, including the base case.

- (c) (4 points) Describe a dynamic program to compute the optimal weight. What are the dimensions of your table? How do you fill it? What is the final value returned? What is the runtime of your algorithm, including sorting and computing all *prior*(*i*) values?

5. **Target or Bust (20 points)** Let n and B be positive integers. You are playing a game that lasts n rounds. You start with a score of 0 and your goal is to reach a score of at least B , for $1 \leq B \leq 3n$. In round i , you can choose one of two actions:

- **SAFE:** you gain exactly 1 point
- **GAMBLE:** you gain 3 points with probability p_i , and 0 points otherwise

You know all $p_i \in [0, 1]$ in advance and the gamble outcomes in each round are independent. You observe the outcome of each round before choosing your action in the next round; your choices may depend on your current score. You want to maximize your probability of getting at least B points in n rounds.

- (a) (3 points) Consider the greedy algorithm which in round i always chooses the action with highest expected point gain. Show this algorithm fails. Your counterexample may consist of just one round.
- (b) (12 points) Give a recurrence relation to return the maximum probability of winning, including base cases. First describe in words the function for your recurrence relation.

- (c) (5 points) Describe a dynamic program to solve this problem based on your recurrence. What are the dimensions of your table? How do you fill it? What is the final value returned? What is the runtime of your algorithm? How can you reconstruct the choices to achieve this probability?