Final Exam

CS 4104 (Summer 2024)

Submit a PDF file containing your solutions on Canvas by 11:59pm on Friday August 5, 2024.

Instructions:

- The Honor Code applies to this homework with the following exception:
  - You are allowed to discuss possible algorithms and bounce ideas with your classmates. Do not discuss proofs of correctness, final answer or running time in detail with your classmate. You must write down your solution individually and independently. Do not send a written solution to your classmates for any reason whatsoever.
  - You are not allowed to consult sources other than your textbook, the slides on the course web page, your own class notes, the TA, and the instructor. **In particular, do not use a search engine or large language models such as ChatGPT**.
- Do not forget to typeset your solutions. Every mathematical expression must be typeset as a mathematical expression, e.g., the square of n must appear as n^2 and not as `n2`. You can use the LaTeX version of the homework problems to start entering your solutions. At the end of each problem are three commented lines that look like the example below. You can uncomment these lines and type in your solution within the curly braces.

```
% \solution{
%
% }
```

- Do not make any assumptions not stated in the problem. If you do make any assumptions, state them clearly, and explain why the assumption does not decrease the generality of your solution
- You must also provide a clear proof that your solution is correct (or a counter-example, where applicable). Type out all the statements you need to complete your proof. You must convince us that you can write out the complete proof. You will lose points if you work out some details of the proof in your head but do not type them out in your solution.
- If you are proposing an algorithm as the solution to a problem, keep the following in mind:
  - Describe your algorithms as clearly as possible. The style used in the text book is fine, as long as your description is not ambiguous. Explain your algorithm in words. A step-wise description is fine. *However, if you submit detailed pseudo-code without an explanation, we will not grade your solutions.*Do not describe your algorithms only for a specific example you may have worked out.
  - Make sure to state and prove the running time of your algorithm. You will only get partial credit if your analysis is not tight, i.e., if the bound you prove for your algorithm is not the best upper bound possible.
  - You will get partial credit if your algorithm is not the most efficient one that is possible to develop for the problem.
- In general for a graph problem, you may assume that the graph is stored in an adjacency list and that the input size is m + n, where n is the number of nodes and m is the number of edges in the graph. Therefore, a linear time graph algorithm will run in $O(m + n)$time.

Problem 1 (10 Points) Estimate an asymptotic bound (Θ) for $3n^4 + 5n^3 + 7$ and prove your estimate is correct using the substitution method.

Problem 2 (5 Points) Prove that $4n^2 \log n + 6n$ is $\Omega(n^2 \log n)$ using the substitution method.

Problem 3 (10 Points) Prove that $5n^2 \log n + 20$ is $\Theta(n^2 \log n)$ using the substitution method.

Problem 4 (25 Points) **Design** and **analyze** a divide and conquer algorithm to find the maximum sum of k non-overlapping subarrays in an array. Each subarray must be contiguous, and the k subarrays must be disjoint.

Example 1:

- Input: $[1, 2, 3, 4, 5, 6, 7, 8, 9]$, $k = 3$
- Expected Output: 24 (Subarrays: $[7, 8, 9]$, $[4, 5, 6]$, $[1, 2, 3]$)

Example 2:

- Input: $[-1, 4, -2, 3, -5, 6, 1, -1, 4, 3, -6]$, $k = 2$
- Expected Output: 14 (Subarrays: $[6, 1, -1, 4, 3]$, $[4]$)

Example 3:

- Input: $[10, -1, 2, 3, -4, 5, -6, 7, 8, -9]$, $k = 2$
- Expected Output: 24 (Subarrays: $[10, -1, 2, 3, -4, 5]$, $[7, 8]$)

Example 4:

- Input: $[1, -2, 3, 4, -5, 6, 7, 8, -9, 10]$, $k = 3$
- Expected Output: 32 (Subarrays: $[6, 7, 8]$, $[10]$, $[3, 4]$)

Problem 5 (25 Points) **Design** and **analyze** a divide and conquer algorithm to find the smallest enclosing circle for a set of points in a 2D plane. The smallest enclosing circle (also known as the minimum bounding circle) is the smallest circle that can completely contain all the given points.

Example 1:

- Input: $[(0, 0), (2, 2), (3, 1), (1, 4)]$
- Expected Output: Circle with center $(1.5, 2)$ and radius 2.5

Example 2:

- Input: $[(1, 1), (2, 5), (5, 2), (6, 6)]$
- Expected Output: Circle with center $(3.5, 3.5)$ and radius 3.5

Example 3:

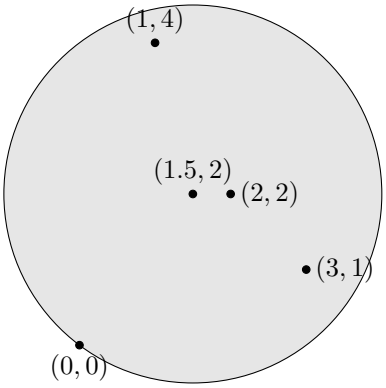
- Input: $[(0, 0), (1, 1), (1, 0), (0, 1)]$
- Expected Output: Circle with center $(0.5, 0.5)$ and radius ≈ 0.707

Example 4:

- Input: $[(1, 1), (2, 2), (3, 3), (4, 4), (5, 5)]$
- Expected Output: Circle with center $(3, 3)$ and radius ≈ 2.828

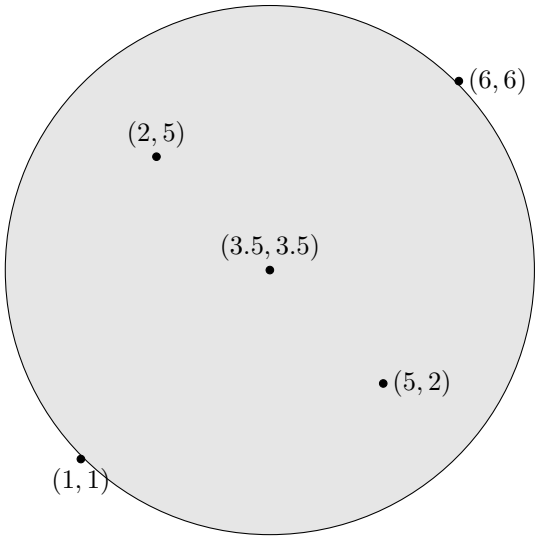
Visual Example 1:

- Input: $[(0, 0), (2, 2), (3, 1), (1, 4)]$
- Expected Output: Circle with center $(1.5, 2)$ and radius 2.5



Visual Example 2:

- Input: $[(1,1),(2,5),(5,2),(6,6)]$
- Expected Output: Circle with center $(3.5, 3.5)$ and radius 3.5



Hint: Consider the properties of the smallest enclosing circle and how it relates to the points on its boundary. When dividing the points, think about how to ensure that the resulting circles from each subset can still combine to form a minimal enclosing circle.

Problem 6 (25 Points) **Design** and **analyze** a divide and conquer algorithm to find the most frequent element in an array with the constraint that the most frequent element appears more than $n/2$ times (majority element). The majority element is guaranteed to exist.

Example 1:

- Input: $[1,2,3,2,2,2,5,2]$
- Expected Output: 2

Example 2:

- Input: $[3,3,4,2,4,4,2,4,4]$
- Expected Output: 4

Example 3:

- Input: $[1,1,1,2,3,4,1,1,1]$

- Expected Output: 1

Example 4:

- Input: $[5, 5, 5, 6, 7, 5, 5, 5, 8, 5]$
- Expected Output: 5

Problem 7 (20 Points) Design a dynamic programming algorithm to find the minimum cost to paint a row of houses such that no two adjacent houses have the same color. Each house can be painted in one of three colors, with a given cost for each color. The costs are provided in a 2D array *costs* where $costs[i][j]$ is the cost of painting house i with color j.

Example 1:

- Input: $[[17, 2, 17], [16, 16, 5], [14, 3, 19]]$
- Expected Output: 10 (Paint house 0 with color 1, house 1 with color 2, and house 2 with color 1)

Example 2:

- Input: $[[7, 6, 2], [5, 8, 7], [3, 4, 6]]$
- Expected Output: 9 (Paint house 0 with color 2, house 1 with color 0, and house 2 with color 1)

Hint: Build up the solution for each house by considering the minimum cost to paint the previous house in each possible color.

Problem 8 (20 Points) Design a dynamic programming algorithm to find the number of ways to decode a message encoded as a string of digits. Each digit or pair of digits maps to a letter ('A' = 1, 'B' = 2, ..., 'Z' = 26).

Example 1:

- Input: "12"
- Expected Output: 2 (Decodings: "AB" and "L")

Example 2:

- Input: "226"
- Expected Output: 3 (Decodings: "BZ", "VF", and "BBF")

Example 3:

- Input: "1234"
- Expected Output: 3 (Decodings: "ABCD", "AWD", and "LCD")

Example 4:

- Input: "301"
- Expected Output: 0 (No valid decodings since '0' cannot be decoded alone and '30' is out of the valid range)

Hint: Consider the number of ways to decode the string by looking at one or two characters at a time.

Problem 9 (15 Points) Design a greedy algorithm to minimize the total waiting time for all customers in a queue. Each customer has a service time, and the goal is to minimize the sum of the waiting times for all customers.

Example 1:

- Input: $[3, 2, 1]$ (Service times of three customers)
- Expected Output: 4 (Serve customers in the order $[1, 2, 3]$)

Example 2:

- Input: $[4, 2, 5, 3]$ (Service times of four customers)
- Expected Output: 10 (Serve customers in the order $[2, 3, 4, 5]$)

Solution: The algorithm sorts the customers by their service times in ascending order and calculates the total waiting time based on this order.

Algorithm 1 Minimize Total Waiting Time

Require: A list of service times T
Ensure: The minimum total waiting time
  Sort T in ascending order
  Initialize $total\_waiting\_time \leftarrow 0$
  Initialize $current\_waiting\_time \leftarrow 0$
  for each time t_i in T **do**
    $total\_waiting\_time \leftarrow total\_waiting\_time + current\_waiting\_time$
    $current\_waiting\_time \leftarrow current\_waiting\_time + t_i$
  end for
  return $total\_waiting\_time$ =0

Time Complexity: $O(n \log n)$ due to sorting.

Rubrics:

- 15 points for correct algorithm

Problem 10 (20 Points) Prove that the Vertex Cover problem can be reduced to the Independent Set problem in polynomial time. Specifically, given an instance of the Vertex Cover problem, design an algorithm to transform it into an instance of the Independent Set problem such that solving the Independent Set problem will yield a solution to the Vertex Cover problem.

Vertex Cover Problem: Given a graph $G = (V, E)$ and an integer k, determine if there exists a subset $V' \subseteq V$ such that $|V'| \leq k$ and every edge in E has at least one endpoint in V'.

Independent Set Problem: Given a graph $G = (V, E)$ and an integer k, determine if there exists a subset $V' \subseteq V$ such that $|V'| \geq k$ and no two vertices in V' are adjacent.