Assignment 5

CS 577 *Divide & Conquer – Geometric Algebraic & Property Searching Applications* Fall 2024

Answer the questions in the boxes provided on the question sheets. If you run out of room for an answer, add a page to the end of the document.

Related Readings: `http://pages.cs.wisc.edu/~hasti/cs240/readings/`

Name: _____ Wisc id: _____

- **Purpose of Homework:**
  - Algorithm design and analysis, like any skill, can only be developed through consistent practice and feedback. Whether it's cooking, playing basketball, integration, gardening, interviewing, or teaching, theoretical knowledge alone is not sufficient. The comfortable feeling of "Oh, sure, I get it" after following a well-presented lecture or hearing a TA explain a homework solution is a seductive, yet dangerous trap. True understanding comes from doing the thing—by actually solving the problems yourself.
  - The homework assignments are your opportunity to practice. Lectures, textbooks, office hours, labs, and guided problem sets are designed to build intuition and provide justification for the skills we want you to develop. However, the most effective way to develop those skills is by attempting to solve the problems on your own. The process is far more important than the final solution.
  - Expect to get stuck. It's normal to have no idea where to start on some problems. That's why you have access to a textbook, lecture slides, and discussions. The journey of wrestling with the problem is an essential part of the learning process.

Scoring Guidelines

You do not need to solve all the exercises!!!

1a) 3.5 points	1b) 3.5 points	1c) 3 points	2a) 12 points	2b) 1 point	2c) 7 points
3a) 3.5 points	3b) 1.5 points	3c) 5 points	4a) 1 point	4b) 2.5 points	4c) 6.5 points
5) 5 points	6a) 5 points	6b) 5 points	6c) 5 points		
7a) 5 points	7b) 5 points	7c) 5 points	8a) 2.5 points	8b) 2.5 points	
8ca) 5 points	8cb) 5 points	8cc) 5 points	8cd) 5 points	8ce) 5 points	
9a) 3.5 points	9b) 3.5 points	9c) 3 points			
10) 5 points					
11a) 2.5 points	11b) 2.5 points	11c) 2.5 points	11d) 2.5 points	11e) 5 points	11f) 5 points
12) 5 points					
13a) 2.5 points	13b) 2.5 points				

- **Total score:** 160 points before rescaling. To pass, you need 100 points. Any extra points are bonus.
- To incentivize better correctness proofs, a multiplier applies to questions $S = \{3, \ldots, 8\}$:
  - Scoring 80% or higher on these gives a 1.5 multiplier.
  - Below 80%, the multiplier is 1.
- Final score formula:

$$\text{Final Score} = (\text{Total Points Earned}) \times \text{Correctness Multiplier}_S$$

- With rescaling, you can score over 200 points, earning extra credit.
- If you only attempt $S = \{3, \ldots, 8\}$, you can earn 127.5 points *(Less than 10 pages)*.

- Keep in mind that the use of AI-generative tools is strictly prohibited. Moreover, be aware that even the most advanced AI systems often produce unsatisfactory or incorrect results, especially when proofs lack rigor. For example, algorithms with nested loops and unjustified n^2 complexity, based on flawed logic, will receive zero points.
- As your professor specializes in Game Theory, you should understand the concept of "common knowledge" — I know that you know that I know. For more information on this concept, see Common knowledge (logic). Proceed accordingly!

Homework Guidelines

- **Collaboration and Academic Integrity:**
  - You are encouraged to work together on homework problems, but you must list everyone you worked with for each problem.
  - You must write everything in your own words and properly cite every external source you use, including ideas from other students. The only sources that you are not required to cite are the official course materials (lectures, notes, homework solutions).
  - Plagiarism is strictly prohibited. Using ideas from other sources or people without citation is considered plagiarism. Copying verbatim from any source, even with citation or permission, is also considered plagiarism. Don't cheat.
- **Submission Instructions:**
  - Submit your homework solutions as PDF files on Gradescope. Submit one PDF file per numbered homework problem.
  - Gradescope will not accept other text file formats such as plain text, HTML, LaTeX source, or Microsoft Word (.doc or .docx).
  - Homework submitted as images (.png or .jpg) will not be graded.
  - Each submitted PDF file should include the following information prominently at the top of the first page: [your full name]_[course title]_[homework assignment number].pdf
- **Solution Writing:**
  - When writing an algorithm, a clear description in English is sufficient. Pseudo-code is not required.
  - Ensure that your algorithm is correct by providing a justification, and analyze the asymptotic running time of your solution. Even if your algorithm does not meet the requested time bounds, you may receive partial credit for a correct, albeit inefficient, solution.
  - Pay close attention to the instructions for each problem. Partial credit may be awarded for incomplete or partially correct answers.

Missions of the Three Musketeers

1. *In the 17th-century kingdom of France, the Three Musketeers—Athos, Porthos, and Aramis—find themselves facing a crucial mission. The life of King Louis XIII is in grave danger, and they must thwart yet another plot by the scheming Cardinal Richelieu. To do so, they need to reach the royal palace in Paris from the distant city of Nice, but they require the three fastest horses from their stable of 25 horses. However, there is a problem: the arena can only fit 5 horses at a time. Luckily, the Musketeers, along with their trusted friend d'Artagnan and his father, can test the horses through multiple races. There's just one catch—timers haven't been invented yet, so they must rely on races alone. Fortunately, the horses have stable performance, meaning that their relative speeds remain consistent in each race.*

   (a) What is the minimum number of races the Musketeers must hold to identify the three fastest horses from the 25? Describe your technique and provide a proof of its correctness.

   (b) Despite arriving in Paris in time to warn the king, the Cardinal, ever devious, has hatched another plot. At the grand gala, where all of Paris will be in attendance, the Cardinal has poisoned one of the 1,000,000 barrels of wine destined for the celebration. The poison is so potent that a single drop can kill anyone who drinks it, but it takes 24 hours for the effects to manifest. As a final mockery, the Cardinal smirks and offers a cruel challenge:

   "There is no poison, my dear Musketeers. But if you're so worried, I generously offer you 100 of my finest mice to taste-test any wine you like... Buahahaha!"

   Acting quickly, the Musketeers hand the Cardinal a glass of wine with a drop from every barrel, forcing him to flee. But the gala will soon begin, and they must find the poisoned barrel before it's too late. With only 24 hours and 100 mice, how can the Musketeers find the poisoned barrel?

   (c) *In the royal vaults of the French palace, a vital secret lies hidden in a special box—the King's most valuable treasure. This box, however, can only be opened with a special key that the Musketeers, Athos, Porthos, and Aramis, now hold in their possession. But the vault is filled with thousands of boxes, all sorted by the size of their keyholes, and only one box can be opened with their key. The Musketeers must find the exact box that their key opens, but time is running short. Each box is arranged in order of keyhole size, from the smallest to the largest. The Musketeers realize they don't have the time to try every box, and they need a faster method to find the correct box before Cardinal Richelieu seizes control of the vault.*

2. Geometry: More Divide and Conquer

3. *Kleinberg, Jon. Algorithm Design (p. 248, q. 5)* Hidden surface removal is a problem in computer graphics where you identify objects that are completely hidden behind other objects, so that your renderer can skip over them. This is a common graphical optimization.

   In a clean geometric version of the problem, you are given n non-vertical, infinitely long lines in a plane labeled $L_1 \dots L_n$. You may assume that no three lines ever meet at the same point. (See the figure for an example.) We call L_i "uppermost" at a given x coordinate x_0 if its y coordinate at x_0 is greater than that of all other lines. We call L_i "visible" if it is uppermost for at least one x coordinate.

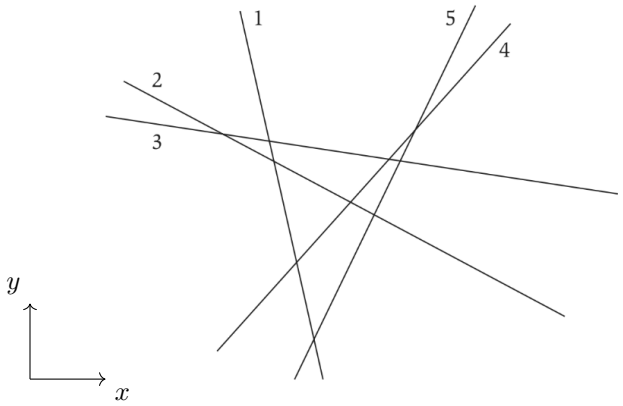

Figure 5.10 An instance of hidden surface removal with five lines (labeled *1–5* in the figure). All the lines except for *2* are visible.

(a) Give an algorithm that takes n lines as input and in $O(n \log n)$ time returns all the ones that are visible.

(b) Write the recurrence relation for your algorithm.

(c) Prove the correctness of your algorithm.

4. In class, we considered a divide and conquer algorithm for finding the closest pair of points in a plane. Recall that this algorithm runs in $O(n \log n)$ time. Let's consider two variations on this problem:

   (a) First consider the problem of searching for the closest pair of points in 3-dimensional space. Show how you could extend the single plane closest pairs algorithm to find closest pairs in 3D space. Your solution should still achieve $O(n \log n)$ run time.

   (b) Now consider the problem of searching for the closest pair of points on the surface of a sphere (distances measured by the shortest path across the surface). Explain how your algorithm from part a can be used to find the closest pair of points on the sphere as well.

   (c) Finally, consider the problem of searching for the closest pair of points on the surface of a torus (the shape of a donut). A torus can be thought of taking a plane and "wrap" at the edges, so a point with y coordinate MAX is the same as the point with the same x coordinate and y coordinate MIN. Similarly, the left and right edges of the plane wrap around. Show how you could extend the single plane closest pairs algorithm to find closest pairs in this space.

Search for a Property

5. We are given an array A of n integers.

   (a) Give a naive algorithm running in $O(n^3)$ time that determines if there are $x, y, z \in A$ (not necessarily distinct entries) such that $x + y + z = 0$.

   (b) Solve the above problem in $O(n^2 \log n)$ time. **Hint:** Consider all sums of two entries.

   (c) Solve the problem in worst-case $O(n^2)$ time and justify its correctness and complexity.

6. *During the height of the First Wizarding War, as Voldemort's forces grew stronger, Albus Dumbledore realized the key to turning the tide lay not in brute force, but in identifying the most powerful witches and wizards who could stand against the Dark Lord. These individuals, known as "Peak Wizards", had innate magical strength far beyond their peers. The wizards stood in a long, winding road leading to Hogwarts, their magical power levels recorded in an array A of length n. Dumbledore knew that somewhere along this path, there were wizards whose power exceeded or matched that of their neighbors — He called them*

the Peak Wizards. Beyond the first and last wizards, the magic faded into oblivion: $A[0] = -\infty$ *and* $A[n+1] = -\infty$*. A Peak Wizard is defined as:*

$$A[i] \geq A[i-1] \text{ (if } i > 1) \text{ \& } A[i] \geq A[i+1] \text{ (if } i < n)$$

With the war escalating and Voldemort's forces closing in, Dumbledore couldn't afford a slow, methodical search. Using his unparalleled wisdom, he has to find a Peak Wizard in $O(\log n)$ *time and justify the correctness*

Coloris Dominus

7. *In one of the most interesting classes at Hogwarts, Professor Flitwick presented a magical problem involving objects of different colors. However, before they could start the exercise, he cast the spell* `Achromatus Totalis` *on the students, causing them to become colorblind! Now, all the objects appeared in shades of gray, making it impossible to compare them based on their colors. However, Professor Flitwick revealed a trick: the shadows of the objects hold the key. If two objects are of the same color, their shadows will be exactly identical. But if they are of different colors, their shadows will be different. The only method they could use to compare the objects was the spell* `Umbra Revelio`*, which would reveal either white sparks for matching shadows or black sparks for differing shadows. The students' task was to find if there was a color among the objects that covered more than half the objects, the so-called* `Coloris Dominus`*, without being able to directly see the colors. They could only compare the shadows.*
Remark:

- *You may assume that the methods of both Hermione and Ron rely on counters they use on the blackboard and the function* `UmbraRevelio`*(x, y), which outputs White Spikes if* $x = y$ *and Black Spikes if* $x \neq y$*.*
- Give emphasis to the correctness of the methods that you will describe.

(a) *Ron, as bored as always, suggested a strategy. He proposed to split the objects into two equal groups and let the others find the majority object in both groups. Then, he would apply* `Umbra Revelio` *between each object in his hand and all other objects. If an object produced more than half white sparks, he would declare its color as* `Coloris Dominus`*. However, his classmates complained that they were doing all the work! Thus, he proposed to reduce their task recursively to other students. According to Hermione, Ron's method would take* $n \log n$ `Umbra Revelio` *spells.*

- ⋆ Describe Ron's method using counters, colleagues and `UmbraRevelio`(x, y) calls
- ⋆ Provide its complexity analysis and correctness.

(b) *But Hermione, ever logical, came up with a smarter solution using a divide-and-conquer approach, but she had a method that would only require* n *spells: (1) The students could pair up the objects into* $n/2$ *pairs. (2) If two objects had the same shadow, they could keep one and discard the other. (3) If they had different shadows, they could discard both. Hermione explained that after each step, the number of objects would be halved. If there was a* `Coloris Dominus` *among the original* n *objects, it would remain in the* $n/2$ *objects, and the process could continue until only one object will survive with* `Coloris Dominus` *the dominant color was found. Finally when the elements are at most* 5*, Hermione applies a brute force comparing every pair. Prove that Hermione was right and that her method finds the* `Coloris Dominus` *is correct using fewer* `Umbra Revelio` *comparisons than Ron's method, actually* $O(n)$ `Umbra Revelio`*.*

- ⋆ Describe Hermione's method using counters, colleagues and `UmbraRevelio`(x, y) calls
- ⋆ Provide its complexity analysis and correctness.

Explain why Hermione's initial algorithm need to apply brute force in case of $n = 3$

(c) *Harry hates corner cases, so he decided to simplify Hermione's algorithm with a non-divide-and-conquer implementation. Harry's algorithm works as follows:*
`He starts with an arbitrary object as the candidate for the Coloris Dominus property and keeps a counter initialized to zero. For each subsequent object, he applies the Umbra Revelio spell: if white sparks appear, he increments the counter; if black sparks appear, he decrements it. If the counter becomes negative, Harry abandons the current candidate and switches to the new object, resetting the counter to zero. At the end of the traversal, Harry is left with a final candidate for the Coloris Dominus property.`

- **What should be the final step in Harry's algorithm to confirm that an object truly possesses the `Coloris Dominus` property?**
- **Why does this algorithm—along with the final step—ensure correctness and run in linear time?**

Bolzano-Weierstrass Theorem and Numerical Analysis

In the following exercises, we will study how to use binary search to solve equations and mathematical problems.

8. A classical theorem in mathematics states that a continuous function has the property that if, for an interval $[a, b]$, we have $f(a) < 0$ and $f(b) > 0$, then there exists a root r in the open interval (a, b) such that $f(r) = 0$.

   (a) Suppose you have an oracle that provides the value of $f(x)$ for any x in constant time. Given an interval $[a, b]$ such that $f(a) < 0 < f(b)$ and $|b-a| = L$, design an algorithm to find an approximate solution x_{solution} such that $|x_{\text{solution}} - x_{\text{root}}| < \epsilon$ in time $O(\log(L/\epsilon))$.

   (b) Prove that if the function f is Lipschitz continuous, i.e., $\forall x, y \in \mathbb{R}$, $|f(x) - f(y)| \leq K|x - y|$ for some constant K, then the previous algorithm will find a solution such that $|f(x_{\text{solution}})| < \epsilon$ in $O(\log(KL/\epsilon))$ steps.

   (c) Suppose you are implementing this algorithm on a real computer system where the maximum representable number is 10^{38}. Consider the numbers $x = 1$ and $y = 10^{38}$. Specifically, if you compute the midpoint as $x_{\texttt{mid}} = \dfrac{x+y}{2}$, this might cause overflow for large values of x and y. Suggest a better approach for computing the midpoint to avoid this overflow.

9. In this exercise, we will explore the discrete case of another classical theorem in mathematics. If we have a continuous function $f([0, 1]) \subseteq [0, 1]$, there exists a fixed point $f(x_\star) = x_\star$ within the interval $[0, 1]$.

   (a) Prove using Bolzano's Theorem that such a point exists.

   (b) Propose an algorithm to compute an approximation of such a fixed point such that $|f(x_{\text{approx}}) - x_{\text{approx}}| \leq \epsilon$ if the function is k-Lipschitz and $f([0, 1]) \subseteq [0, 1]$.

   (c) Let's analyze the discrete version of the previous theorem. Let k and n be positive natural numbers, with $k < n$. Consider an array $A[0 \ldots n]$, which contains $n + 1$ natural numbers, each of which is smaller than or equal to n. We assume that for any two consecutive elements in the array, the absolute difference between their values is at most k. Formally, for every j: $|A[j] - A[j + 1]| \leq k$ (this is a discrete version of the Lipschitz continuity property in discrete time).

      (a) Prove the following lemma: ***Lemma:*** *There must exist two consecutive indices $\{i_\star, i_\star - 1\}$ such that (a) $i_\star - A[i_\star] \geq 0$ and (b) $(i_\star - 1) - A[i_\star - 1] \leq 0$.*

      (b) Show that there exists an index j such that: $|A[j] - j| \leq \frac{k+1}{2}$.

      (c) Design an efficient algorithm to find an index j such that $|A[j] - j| \leq \frac{k+1}{2}$. Specify the computational complexity of your algorithm and justify its correctness.

(d) Describe the recurrence of your divide & conquer algorithm and specify the computational complexity of your algorithm .

(e) Justify the correctness of your algorithm.

Algebraic Improvement

10. In the following exercise we study algorithms for multiplication of two polynomials $P(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$ and $Q(x) = b_0 + b_1x + b_2x^2 + \cdots + b_nx^n$ given as arrays $p[n] = [a_0, \cdots, a_n]$

    (a) Show how to compute the quadratic polynomial by multiplying two linear polynomials $ax + b$ and $cx + d$ using only 3 multiplications.

    (b) Provide a divide & conquer algorithm for multiplying polynomials of degree n with a time complexity of $\Theta(n^{\log_2 3})$.
    [Hint] Find a way to group the coefficients of the polynomial (e.g., based on high and low powers, or depending on whether the exponent is odd or even) so that the idea from the previous question can be applied.

    (c) Describe the recurrence of your divide & conquer algorithm and specify the computational complexity of your algorithm.

11. *Erickson, Jeff. Algorithms (p. 58, q. 25 d and e)* Prove that the following algorithm computes $\gcd(x, y)$ the greatest common divisor of x and y, and show its worst-case running time.

```
BINARYGCD(x,y):
if x = y:
    return x
else if x and y are both even then return 2*BINARYGCD(x/2,y/2)
else if x is even then return BINARYGCD(x/2,y)
else if y is even then return BINARYGCD(x,y/2)
else if x > y then return BINARYGCD( (x-y)/2,y )
else return BINARYGCD( x, (y-x)/2 )
```

12. *Dasgupta et Al., Algorithms (p. 83-84, ex 2.26-2.27)* In the following exercise, we will analyze the connection between the problem of square a number/matrix and its multi cations.

    (a) Let the square of a matrix A is its product with itself, AA. Show that five multiplications are sufficient to compute the square of a 2×2 matrix.

    (b) What is wrong with the following algorithm for computing the square of an $n \times n$ matrix?
    "Use a divide-and-conquer approach as in Strassen's algorithm, except that instead of getting 7 subproblems of size $n/2$, we now get 5 subproblems of size $n/2$ thanks to part (a). Using the same analysis as in Strassen's algorithm, we can conclude that the algorithm runs in time $O(n^{\log_2 5})$.

    (c) In fact, squaring matrices is no easier than matrix multiplication. In this part, you will show that if $n \times n$ matrices can be squared in time $S(n) = O(n^c)$, then any two $n \times n$ matrices can be multiplied in time $O(n^c)$. Given two $n \times n$ matrices A and B, show that the matrix $AB + BA$ can be computed in time $3S(n) + O(n^2)$.

    (d) Given two $n \times n$ matrices X and Y, define the $2n \times 2n$ matrices A and B as follows:

$$A = \begin{bmatrix} X & 0 \\ 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & Y \\ 0 & 0 \end{bmatrix}$$

    What is $AB + BA$, in terms of X and Y?

    (e) Using the previous parts, argue that the product XY can be computed in time $3S(n) + O(n^2)$. Conclude that matrix multiplication takes time $O(n^c)$.

(f) Professor F. Lake tells his class that squaring an n-bit integer is asymptotically faster than multiplying two n-bit integers. Should they believe him?

13. *Dasgupta et Al., Algorithms (p. 84, ex 2.28)* The Hadamard matrices $H_0, H_1, H_2, \ldots$ are defined as follows:

- H_0 is the 1×1 matrix $\begin{bmatrix}1\end{bmatrix}$.
- For $k > 0$, H_k is the $2^k \times 2^k$ matrix:

$$H_k = \begin{bmatrix} H_{k-1} & H_{k-1} \\ H_{k-1} & -H_{k-1} \end{bmatrix}$$

Show that if v is a column vector of length $n = 2^k$, then the matrix-vector product $H_k v$ can be calculated using $O(n \log n)$ operations providing the necessary recurrence and justification. Assume that all the numbers involved are small enough that basic arithmetic operations like addition and multiplication take unit time.

14. *Dasgupta et Al., Algorithms (p. 84, ex 2.29)* Suppose we want to evaluate the polynomial $p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$ at point x.

(a) Show that the following simple routine, known as Horner's rule, does the job and leaves the answer in z:

$$\begin{cases} z = a_n \\ \text{for } i = n-1 \text{ down to } 0: \\ z = zx + a_i \end{cases}$$

(b) How many additions and multiplications does this routine use, as a function of n?