Algorithm Design and Analysis
Assignment 1
Deadline: Oct 17, 2024

1. (25 points) Asymptotic notations.

   (a) In each of the following situations, indicate whether $f = O(g)$, or $f = \Omega(g)$, or both (in which case $f = \Theta(g)$). Justify your answer.

      1. $f(n) = n^{1/2}$ and $g(n) = 5^{\log_2 n}$
      2. $f(n) = 100n + \log n$ and $g(n) = n + (\log n)^2$
      3. $f(n) = (\log n)^{\log n}$ and $g(n) = n/\log n$
      4. $f(n) = (\log n)^{\log n}$ and $g(n) = 2^{(\log_2 n)^2}$
      5. $f(n) = \sum_{i=1}^{n} i^k$ and $g(n) = n^{k+1}$

   (b) Let $f(n) = \left(2 \cdot \lceil \frac{n}{2} \rceil\right)!$ and $g(n) = \left(2 \cdot \lfloor \frac{n}{2} \rfloor + 1\right)!$. Prove that neither $f = O(g)$ nor $f = \Omega(g)$ is true.

2. (25 points) Prove the following generalization of the master theorem. Given constants $a \geq 1, b > 1, d \geq 0$, and $w \geq 0$, if $T(n) = 1$ for $n < b$ and $T(n) = aT(n/b) + n^d \log^w n$, we have

$$T(n) = \begin{cases} O(n^d \log^w n) & \text{if } a < b^d \\ O(n^{\log_b a}) & \text{if } a > b^d \\ O(n^d \log^{w+1} n) & \text{if } a = b^d \end{cases}.$$

3. (25 points) For two vectors $\mathbf{a} = (a_1, \ldots, a_d), \mathbf{b} = (b_1, \ldots, b_d) \in \mathbf{R}^d$, we say $\mathbf{a}$ is *greater than* $\mathbf{b}$ if $a_k > b_k$ for each $k = 1, \ldots, d$. You are given two collections of vectors $A, B \subseteq \mathbb{R}^d$. The objective is to count the number of pairs $(\mathbf{a}, \mathbf{b}) \in (A, B)$ such that $\mathbf{a}$ is greater than $\mathbf{b}$. You can assume all the entries in all the vectors are distinct. Let $n = |A| + |B|$ and d be the dimension of the vectors.

   (a) Design an $O(n \log n)$ time algorithm for this problem with $d = 1$.

   (b) Design an algorithm for this problem with $d = 2$. Your algorithm must run in $o(n^{1.1})$ time.

   (c) Generalize the algorithm in part (b) so that it works for general d. Analyze its running time. The running time must be in terms of n and d.

4. (25 points) Let A be a square matrix. This question discusses the computation of A^2.

   (a) 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 $O(n^c)$.

   1. Given two $n \times n$ matrices A and B, show that the matrix $AB + BA$ can be computed in time $3S(n) + O(n^2)$.
   2. Given two $n \times n$ matrices X and Y, define the $2n \times 2n$ matrices A and B as follows:
   $$A = \begin{bmatrix} X & 0 \\ 0 & 0 \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} 0 & Y \\ 0 & 0 \end{bmatrix}.$$
   What is $AB + BA$, in terms of X and Y?
   3. Using 1 and 2, argue that the product XY can be computed in time $3S(2n) + O(n^2)$. Conclude that matrix multiplication takes time $O(n^c)$.

5. How long does it take you to finish the assignment (including thinking and discussion)? Give a score (1,2,3,4,5) to the difficulty. Do you have any collaborators? Please write down their names here.