Homework 1

CS6033 Design and Analysis of Algorithms I

Fall 2024

(Sec. B, Prof. Yi-Jen Chiang)

Due: Wed. 9/25 by 1pm
(submit online on NYU Brightspace; one submission per group)
Maximum Score: 100 points

Note: This assignment has 2 pages.

1. (25 points)
In the following, let f and g be positive increasing functions. Answer each question and **briefly justify your answers.** You get no points if you do not give a justification.

(a) Given that $f(n) = \Omega(g(n))$, is it possible that $f(n) = O(g(n))$? Is it always true that $f(n) = O(g(n))$? **(5 points)**

(b) Given that $f(n) = O(g(n))$, is it possible that $f(n) = o(g(n))$? Is it always true that $f(n) = o(g(n))$? **(5 points)**

(c) Given that $f(n) = \Theta(g(n))$, is it possible that $f(n) = \Omega(g(n))$? Is it always true that $f(n) = \Omega(g(n))$? **(5 points)**

(d) Is it possible that $f(n) = o(g(n))$ and also $f(n) = \Omega(g(n))$? **(5 points)**

(e) Is it possible that $f(n) + g(n) = O(\min(f(n), g(n)))$? Is it always true that $f(n) + g(n) = O(\min(f(n), g(n)))$? **(5 points)**

2. (6 points)
Give two **asymptotically different** functions, each of which belongs to both $\omega(n)$ and $o(n^2)$. Briefly justify your answers.

3. (20 points)
Problem 3-2 of the Textbook (p.71), **(c) - (f)**. Note that "A is O of B" means $A = O(B)$, and similarly for other notations ($o, \Omega, \omega, \Theta$). In your write up, copy and fill up the table with "Yes" or "No" in each table entry; after the table, **briefly justify your answer** for each sub-question. You get no points if you do not give a justification.
(**Note: 5 points for each of (c) - (f).**)

4. (24 points)
Let $S_1(n) = \sum_{k=1}^{n} k$, $S_2(n) = \sum_{k=1}^{n} k^2$ and $S_3(n) = \sum_{k=1}^{n} k^3$. In class, we already showed how to calculate $S_1(n)$ and $S_2(n)$. Your task here is to apply similar methods for $S_3(n)$.

(a) Without calculating the closed form, use rough estimations to derive a lower bound and an upper bound for $S_3(n)$, so that you can use them to express $S_3(n)$ in the $\Theta()$ notation. Give this $\Theta()$ notation in the **simplest form** (e.g., $\Theta(n)$ is in the simplest form but $\Theta(2n)$ is not). **(6 points)**

(b) Use the *perturbation method* (as discussed in class) to derive the closed form for $S_3(n)$.
(**Background Information:** You would need the formula of $S_1(n)$ and $S_2(n)$, which we already know: $S_1(n) = n(n+1)/2$ and $S_2(n) = n(n+1)(2n+1)/6$.) **(18 points)**

5. (25 points)
Recall that a classical *tower of Hanoi* game is as follows: There are 3 rods A, B, C and n disks, where the disks are all in *distinct* sizes. Initially all disks are on rod A in the order of *increasing sizes*, the smallest on the top. The objective is to move all disks from rod A to rod B (where the disks in B are also in the order of increasing sizes), with the following rule: Each move takes the top disk from one rod and places that disk to the top of another rod, with the restriction that no larger disk can be placed on top of a smaller disk.
(It is well-known that we can solve it recursively: First recursively move the top $n-1$ disks from A to C, then move the last, largest disk from A to B, and finally recursively move the $n-1$ disks from C to B. If T_n is the minimum number of moves to complete the task for n disks, then we have $T_n = 0$ when $n = 0$ and $T_n = 2T_{n-1} + 1$ when $n > 0$.)

Now consider the **cyclic tower of Hanoi** game: it is the original tower of Hanoi game **with an additional restriction**: a disk can only be moved from A to B, or from B to C, or from C to A. **Under the new rules**, let Q_n be the minimum number of moves needed to transfer a tower of n disks from A to B, and R_n the minimum number of moves needed to transfer a tower of n disks from B to A. Prove that

$$Q_n = \begin{cases} 0 & \text{if } n = 0, \\ 2R_{n-1} + 1 & \text{if } n > 0, \end{cases} \quad \text{and}$$

$$R_n = \begin{cases} 0 & \text{if } n = 0, \\ Q_n + Q_{n-1} + 1 & \text{if } n > 0. \end{cases}$$

(**Note:** You do **not** need to solve these recurrences.)
(**Hint:** Think about new recursive algorithms.)