CS3230: Assignment for Week 2

Due: Monday, 4th Sep 2023, 6 pm SGT.

Please upload a PDF file containing your solution (hand-written & scanned, or typed) by 4th Sep, 6 pm on Canvas. You may discuss the problems with your classmates, though you should write up your solutions on your own. Please note the names of your collaborators in your submission. You may want to refer to the plagiarism policy from Lecture 2.

Problems for Submission

1. **[3 points]**

   Prove the following properties by definition:

   (a) (Transitivity) $f(n) = O(g(n))$ and $g(n) = O(h(n)) \implies f(n) = O(h(n))$

   (b) (Symmetry) $f(n) = \Theta(g(n)) \iff g(n) = \Theta(f(n))$

2. **[4 points]** Below are eight functions $f_1, f_2, f_3, f_4, f_5, f_6, f_7$ and f_8.

   - $f_1(n) = n^3 + 3n + \sin(n)$
   - $f_2(n) = \left(2^{2^{\lg \lg n}}\right)^2$
   - $f_3(n) = \sum_{i=1}^{n^3} \frac{in^3}{3^i}$
   - $f_4(n) = (\lg \lg n)^n$
   - $f_5(n) = n^2 \log_n n!$
   - $f_6(n) = 2^{\lg \lg n}$
   - $f_7(n) = 3230n - \lg \lg n$

- $f_8(n) = n^{\sum_{i=1}^{\infty} \frac{1}{3^i}}$

(a) Order these functions asymptotically, from the smallest to the largest. To simplify notations, we write $f(n) \leq g(n)$ to mean $f(n) = O(g(n))$ and $f(n) \approx g(n)$ to mean $f(n) = \Theta(g(n))$. For example, the four functions $n^2, n, n^2 + n$ and n^3 could be sorted in increasing order of growth as follows: $n \leq n^2 \approx n^2 + n \leq n^3$. Proofs are not required in this part.

(b) Prove rigorously the relationship between these pairs:

i. f_6 and f_7,

ii. f_5 and f_1.

This means, for example, if you believe that $f_6(n) = \Theta(f_7(n))$, then prove it rigorously.

3. **[3 points]** Alice and Bob have proposed 2 algorithms for finding the $\log(n)$ nearest neighbours of a specific point, P, on a graph. Both algorithms are explained below. There are n number of points and the time taken to calculate the distance between two points, i and j, $FindDist(i, j)$ is $O(n^2)$.
Note that S and D are arrays; that means operation time on them is $O(1)$.

Algorithm by Alice

(a) Initialize $S[i] = 0$ for all the points. Here $i \in \{1, 2, 3, ..., n\}$

(b) For each point, compute $D[i]$. Here $D[i]$ denotes the distance between point i and P.

(c) Iterate $\log(n)$ number of times through all the points to do the following procedure in every iteration. Find the smallest $D[i]$ with the condition $S[i] = 0$. After the full scan through all the points, mark $S[min] = 1$. Here min is the location where $D[min]$ is smallest and $S[min] = 0$.

(d) Return $\log(n)$ points with indices where $S[i] = 1$.

Algorithm by Bob

(a) Initialize S[i] = 0 for all the points. Here i $\in$ {1, 2, 3, ..., n}

(b) Iterate $\log(n)$ number of times through all the points to do the following procedure in every iteration. Find the distance between each point and P. This steps are only done for the locations is' with $S[i] = 0$. The minimum distance location will be marked with $S[min] = 1$.

(c) Return $\log(n)$ points with indices where $S[i] = 1$.

Find the running time of each algorithm in n. Whose algorithm do you think runs faster? Briefly justify your answer.