

CSE 101 First Homework (Homework 1)
Spring 2013
Background, (Order and Recurrence Relations, simple algorithms)
Due Tuesday, April 9
100 points total

Analyzing algorithms, 20 pts. each Short answer, but please give a clear justification for all claims.

Assume $\text{proc}(I)$ is an algorithm that takes $\Theta(I)$ time and does not change I . What are the orders of the running times of the following two algorithms?

Alg1(n)

1. begin;
2. $I \leftarrow 1$;
3. While $I \leq n$ do:
4. begin;
5. $\text{proc}(I)$
6. $I++$
7. end;
8. end;

Alg2(n)

1. begin;
2. $I \leftarrow 1$;
3. While $I \leq n$ do:
4. begin;
5. $\text{proc}(I)$
6. $I \leftarrow 2 * I$
7. end;
8. end;

Order Notation, 5 pts. each Is $2^{\lceil \log n \rceil} \in O(n)$? $\in \Omega(n)$? Why or why not?
(When unspecified, logs are base 2).

Is $\log(n!) \in O(n \log n)$? Why or why not?

Is $4^n \in O(2^n)$? Why or why not?

If f and g are functions from positive integers to positive integers, is $f(n) + g(n) \in \Theta(\max(f(n), g(n)))$?

Summing triples (20 points) Let $A[1, \dots, n]$ be an array of positive integers.

A *summing triple* in A is 3 distinct indices $1 \leq i, j, k \leq n$ so that $A[i] + A[j] = A[k]$. Give and analyze an algorithm that, given A , determines whether there is any summing triple in A . Your algorithm must take $o(n^3)$ time, i.e., some time function that is asymptotically strictly faster than $O(n^3)$.

Implementation (20 points) Implement the algorithm you gave for the summing triples problem above. For n as many different powers of two as possible, and for many random arrays where each of the n elements $A[i]$ has a random value between 1 and n , try your algorithm and plot the average time your algorithm took on a log-log scale, i.e., plot $\log(n)$ on the x -axis and $\log(\text{Time})$ on the y -axis. Then plot the same information but where each $A[i]$ has a random value between 1 and n^6 . Do you see a difference? If so, can you explain it?