

15-251: Great Theoretical Ideas In Computer Science

Recitation 10

Big Oh!

1. Which is asymptotically greater: $\log(\log^* n)$ or $\log^*(\log n)$?
2. Prove or disprove:
 - $f(n) + O(f(n)) = \Theta(f(n))$
 - $f(n) = O(f(n/2))$
 - $f(n) + g(n) = O(\min(f(n), g(n)))$
 - $f(n) = O(f(n)^2)$
3. If $f(n) = O(g(n))$, is $g(n) = O(f(n))$?

Algorithms

4. Sub-linear Algorithms:
 - Given an array of data, when can we perform a sub-linear search of the data?
 - Given an unsorted array of data, can we sort it in sub-linear time?
5. Selection Sort: Given an array of n elements, traverse it to identify the smallest, and then recursively sort the remaining array. What is the recurrence $T(n)$ for selection sort?
6. Merge Sort: Given an array of $n = 2^k$ numbers, merge sort splits the array into two halves, recursively sorts both halves, and merges them. If the time taken to merge is just the size of the halves, write down the recurrence for $T(n)$.

General Recurrences

Consider the following type of recurrence:

$$\begin{aligned}T(n) &= aT(n/b) + cn^k \\T(1) &= c\end{aligned}$$

for positive constants a, b, c, k . This recurrence corresponds to the time spent by an algorithm that divides the problem into a pieces of size n/b , solving each one recursively, and then doing cn^k work stitching them together.

7. If you are told that $a < b^k$, then can you find a Theta bound for $T(n)$? What happens if $a = b^k$? For simplicity, assume n is a power of b .
8. Give the values for a, b , and k if the algorithm under consideration is merge sort. What Theta bound can you find for merge sort?
9. How about the binary search algorithm for identifying if an element is present in a sorted array?
10. Prove that, if $T(n) = n^5 + T(n - 1)$, and $T(1) = 1$, then $T(n) = \Theta(n^6)$.