

CS325 Assignment #1, Summer 2009, OSU

This assignment is due on Thursday, 7/2/2009. You are required to **type your solution**, and turn it in printed at the beginning of class.

1. Prove by induction that $\sum_{i=0}^n r^i = \frac{1-r^{n+1}}{1-r}$. This is called the geometric series.
2. Prove that $2n^7 + n^5 + 3 \in O(n^7)$ using the definition of O , i.e. find c and n_0 (do not use the limit trick or any of the property shortcuts).
3. Take the following list of functions and arrange them in ascending order of growth rate. Prove that your answer is correct, i.e. if you are saying that f_1 comes before f_2 , then you must prove $f_1 \in O(f_2)$ (Hint: using the limit trick is probably the easiest way to do most of the proofs).

$$f_1(n) = n^{2.5}$$

$$f_2(n) = \sqrt{2n}$$

$$f_3(n) = n + 10$$

$$f_4(n) = 10^n$$

$$f_5(n) = 100^n$$

$$f_6(n) = n^2 \log(n)$$

4. Prove that $\log(\log n) \in O(\log n)$ using either the limit trick or from the definition of big- O .
5. Textbook, Exercise 2.5, parts (a)-(e). Use the Master Theorem to solve the following recurrences:

(a) $T(n) = 2T(n/3) + 1$

(b) $T(n) = 5T(n/4) + n$

(c) $T(n) = 7T(n/7) + n$

(d) $T(n) = 9T(n/3) + n^2$

(e) $T(n) = 8T(n/2) + n^3$

6. For each of the following algorithms, state its best-case and worst-case runtime:

(a) Selection Sort

(b) Bubble Sort

(c) Merge Sort

(d) Quick Sort

(e) Heap Sort

(f) Radix Sort

7. You have data on the price of a stock for n days. The price is $p(i)$, for $i = 1, \dots, n$. Suppose that during these n days, you can buy the stock once, then sell it again later. In this question, you will consider algorithms to find the day to buy and the day to sell that maximizes your profit. More precisely, suppose you buy on day a and sell on day b , then your goal is to find a and b that maximize $p(b) - p(a)$.

- (a) Give pseudocode for an iterative algorithm to solve this problem with $O(n^2)$ runtime.
 - (b) Give pseudocode for a recursive divide and conquer algorithm with $O(n \log n)$ runtime. Show why your algorithm has this runtime. Prove that it is correct by induction.
8. You have a pile of identical jars, and a ladder with n rungs. You can climb some number of rungs up the ladder, then drop a jar, and it may or may not break. You want to find the highest rung from which you can drop a jar and not have it break. We call this the highest safe rung.

Suppose can break at most 2 jars. Give a strategy for finding the highest safe rung that requires you to drop a jar at most $f(n)$ times, for some function $f(n)$ that is $o(n)$. Note this is little- o , so your approach must be better than $O(n)$. Hint: there are several possible runtimes that are better than $O(n)$.

9. Textbook, Exercise 2.12
10. Textbook, Exercise 2.17