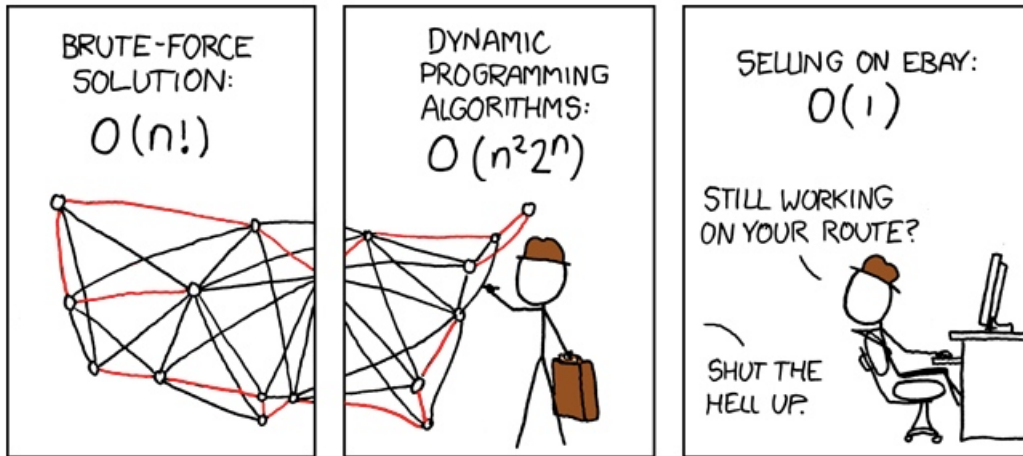


# CS302 - Assignment 2

Due: Tuesday, Feb. 21 at the beginning of class

Hand-in method: paper



<http://xkcd.com/399/>

1. (6 points) The table below contains *actual* run times for 6 different algorithms. The input sizes ranged from 1000 to 32000 seen at the top of the table. For each of the algorithms, give the  $\theta$  complexity of the algorithms based on the running times and include a brief explanation for your answer.

Algorithm	1000	2000	4000	8000	16000	32000
$A_1$	50	378	3,345	26,300	215,680	1,658,002
$A_2$	99	110	105	976	103	100
$A_3$	60	130	237	501	954	1999
$A_4$	1005	1095	1201	1289	1420	1540
$A_5$	5	21	84	311	1304	5280
$A_6$	10	22	50	108	245	533

2. (8 points) Arrange the functions below in ascending order of growth rate. Specifically, if  $f(n) = O(g(n))$  then  $f(n)$  should be before  $g(n)$  in the list. If two functions are asymptotically equal, i.e.  $f(n) = \Theta(g(n))$  then note this in the list by including all elements in a set. For example, given:  $n, \log n, n + 4,$  and  $n^2$  the list would be:  $\log n, (n, n + 4), n^2$ .

$$\begin{array}{cccc}
 \left(\frac{3}{2}\right)^n & 7 & 5n + 20 & 3^{3^n} \\
 n & 3^n & n3^n & n^5 \\
 n! & 3\sqrt{\log n} & n^{\log n} & \frac{1}{2}n \log(n + 5) \\
 10^6 & n^n & (3n)^2 & \log \log n
 \end{array}$$

3. (15 points) Big O

For each of the statements below, state whether it is **true** or **false** and then *prove* your answer.

- (a)  $15n^3 \log n + 10n^2 + 50$  is  $O(n^3 \log n)$ .
- (b)  $3n^2 - 12n + 2$  is  $\Omega(n^3)$
- (c)  $2^{n+1}$  is  $\Theta(2^n)$
- (d)  $2^{2n}$  is  $O(2^n)$
- (e)  $\log(n!)$  is  $O(n \log n)$  (Hint: compare  $n!$  and  $n^n$ )

4. (5 points) Insertion sort

Inspired by all the cool new sorting algorithms you heard about in class, you decide to come up with a cool new sorting algorithm called *your\_name\_here*sort. You notice that *Insertion - Sort* seems inefficient in how it finds the correct location to insert the next value. You decide that rather than linearly searching one at a time to find the correct place, you're going to use binary search to find the correct place. Is this an improvement? If yes, state the running time of this new algorithm. If no, explain why this is not an improvement. Be specific and clear.

5. For the following problems, write **pseudocode** solutions and *state the worse case running time* (in terms of  $\Theta$  or  $O$  where appropriate). You will be graded on the efficiency of your solutions.
- (a) (5 points) Given two lists of numbers  $A$  and  $B$  of lengths  $m$  and  $n$  respectively, return the intersection of the lists, i.e. all those

numbers in  $A$  that also occur in  $B$ . You can use procedures that we've discussed in class, but no others (e.g. no hashtables). You can assume that in any given list, the numbers are unique.

- (b) (10 points) Given a *sorted* list of unique integers  $A[1..n]$ , determine if an entry exists such that  $A[i] = i$ . If an entry exists, return the index, otherwise, return *null*. (Hint: You can do better than  $O(n)$ . Think divide-and-conquer.)

### Just for fun

6. (1 brownie point) Given two *sorted* arrays  $A$  and  $B$  of lengths  $m$  and  $n$  respectively, return the  $k$ th smallest element in the union of the two lists. Your runtime should be in terms of both  $m$  and  $n$  and should not depend on  $k$ .