Dylan Mackey

CS251 - Homework 1: Algorithm Analysis

Out: January 15, 2016 @ 9:00 pm
Due: January 22, 2016 @ 9:00 pm

Important: Each question has only one correct answer. Additionally, you must provide an explanation on each question. Any choice without an explanation, even though it is correct, will be graded with 0 points.

1) Select the **tightest** *big-Oh* expression for the following expressions:

1-1) $1723689n\ log(n^2) + 768^{100}$ constant

a. $O(n\ log(n^2))$
b. $O(768^{100})$
c. $O(n\ log(n))$
d. $O(log(n))$

768^{100} is a constant, so it can be ignored. Therefore, we only include $1723689n\ log(n^2)$, which is simplified to $2n\ log(n)$, so $O(n\ log(n))$

1-2) $2^n + n!$

a. $O(2^n)$
b. $O(2^n + n!)$
c. $O(n!)$
d. $O(2^n\ n!)$

$n!$ grows faster than 2^n, thus the big Oh is $O(n!)$

2) Select the tightest *big-Oh* expression for the following pseudo codes:

2-1)

```
int sum = 0;
for( i = 0; i < n; i++) ∩
  for( j = 0; j < i; j++) ∩
    for( k = 0; k < j; k++) ∩
      sum+=sum;
```

a. $O(n^3)$
b. $O(n^2 log(n))$
c. $O(i^n\ j^n)$
d. $O(n^3 log(n))$

sum += sum is executed at most $n \cdot n \cdot n$ times, so $O(n^3)$

$1, 2 : 2 \cdot 2 : 4 \cdot 2 : 8$

2-2)

```
int prod = 2;
for(int i = 1; i <= n; i*=2)
  for(int j = 1; j <= n^2; j++)
    prod*=prod;
```

log(n) — since i is multiplied by 2 each time, this is executed log(n) time.

n^2 — Upper bound is n^2, so @most n^2 times,

a. $O(n^3)$
b. $O(n^2 log(n))$
c. $O(prod^n\ j^n)$
d. $O(prod^3\ log(n))$

$O(n^2 \log(n))$ is the max # of times executed.

2-3)

```
for( i = 0; i < n; i++)
  for( j = 1; j < i; j *= 2)
    O(k) work where k is a constant
```

Executed n times

$\log(n)-1$ — j starts @ 1 and is incremented by x2, so $\log(n)-1$

a. $O(n^3)$
b. $O(n\ log(n))$
c. $O(n\ log(n)\ j^k log(n))$
d. $O(n^3 log(n))$

$n(\log(n)-1)$
$n\log(n) - n$
$= O(n \log(n))$

3) Find the *big-Θ* expression for the following *big-Oh* and *big-Ω* expressions:

3-1) $O(3n^2\ log(n) - n)$

a. $\theta(n^2 log(n))$
b. $\theta(n)$
c. $\theta(n^2)$
d. $\theta(n\ log(n))$

$3n^2 \log(n) - n$
$\leq 3n^2 \log(n) + n^2 \log n$
$\leq 4n^2 \log(n)$
$\leq \Omega\ n^2 \log(n)$

3-2) $\Omega(\sum_{i=0}^{n} i^2)$

a. $\theta(n)$
b. $\theta(n\ log(n))$
c. $\theta(n^2)$
d. $\theta(n^3)$

$\sum i^2 = \frac{n(n+1)(2n+1)}{6}$ ← Summation equation

$= \frac{(n^2+n)(2n+1)}{6}$

$= \frac{2n^3+n^2+2n^2+n}{6}$

$= \frac{2n^3+3n^2+n}{6}$

$= \theta(n^3)$

4) What is the **optimal** amount of work to find a name in the phone book in term of the number of entries in the phonebook (n)? Note: Phonebook is already sorted.

a. $log(n)$
b. n
c. $n\,log(n)$
d. 1

By using the binary search method, we would return a worst case scenario of $O(log(n))$.

5) Using a linear time searching algorithm, what is the longest time it might take to find a particular page in all pages that Google searches in one day? assuming it counts 2,670,286,179 pages per day.

a. 2,670,286,179
b. 3,000,000,000
c. 2,670,286,179 / 2
d. 2 * 2,670,286,179

linear equation, so $O(n)$, where $n = 2,670,286,176$, so the longest time is 2,670,286,176

6) You wish to perform a daily re-sorting of the most visited student websites from CS251. You are allowed to consider the following approaches (in all of them *n* is the number of students):

Approach A) If you assume that the popularity of the websites among students do not change much from day to day, you can implement an $O(n)$ average running time algorithm to keep the website popularity sorted in daily manner (*n* is the number of students who search the website on that day). This algorithm costs 100 time units per step to perform. $Cost = 100n$

Approach B) If you ignore the assumption of constant popularity, you can be provided with an $O(n^2)$ sorting algorithm to keep the website popularity sorted in daily manner (based on the search of student on that day). This algorithm costs 5 time units per step to perform. $Cost = 5n^2$

Approach C) Assume you are provided with an $O(n\,log(n))$ sorting algorithm but we do not know the cost of this algorithm per unit. However, we know that 100 students are registered to CS251. $cost = 100\,log(100)$

Note: Base of the logarithm is 10 for all the calculations.

6-1) How many students should register to CS251 so Approaches A and B have the same *big-Oh* time complexity?

a. 200
b. 30
c. ~~25~~
d. 20

$100n = 5n^2$

$20 = n$

6-2) What should be the cost per time unit for Approach C so it behaves the same as Approach A?

a. 1
b. 50
c. 100
d. 105

$n=100$, plug in to equation

$100(100) = 100 \times \log(100)$

$100 = x \log(100)$

$x = \frac{100}{\log(100)}$

$x = 50$

Submit Instructions:

The homework must be turned in by the due date and time using the turnin command. Follow the next steps:

1. Login to data.cs.purdue.edu (you can use the labs or a ssh remote connection).
2. Make a directory named with your username and copy your solution (**in pdf format**) there.
3. Go to the upper level directory and execute the following command:
   turnin -c cs251 -p hw1 your_username
   (**Important:** previous submissions are overwritten with the new ones. Your last submission will be the official and therefore graded).
4. Verify what you have turned in by typing **turnin -v -c cs251 -p hw1**
   (**Important:** Do not forget the **-v** flag, otherwise your submission would be replaced with an empty one).