Connor Brown

CS251 - Homework 1: Algorithm Analysis

Out: January 12, 2018 @ 9:00 pm

Due: January 19, 2018 @ 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) $1723689n\ log(n^2) + 768^{100}$

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

The answer is A. This is because the dominant term within an expression is what is used for the big-O notation.

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

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

The answer is A. This is because the dominant term within an expression is what is used for the big-O notation.

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

2-1)

```
int mult = 1;                          1
for( i = 0; i < n; i++)                n
  for( j = 0; j < i; j++)              n^2
    for( k = 0; k < j; k++)            n^3
```

mult *= 2; 1

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

The answer is B, because after analysis of each line the highest order is n^3. This is because with each iteration of the for loop it runs for n^(level of loops) so three loops becomes n^3.

2-2)

int sum = 0; **1**

for(int i = 1; i <= n; i*=2) **n**

for(int j = 1; j <= n^2; j++) **n^3**

sum += sum; **1**

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

The answer is A. Normally with each for loop there is an additional power to n added, however because the second for loops to n^2 it be n^2 for every n in i. Which makes the like take n^3 to iterate.

2-3)

for(i = 0; i < n; i++) **n**

for(j = 1; j < i; j *= 2) **n log(n)**

O(k) work where k is a constant **1**

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

The answer is B. This is because the second for loop doesn't run to n but instead j less than i, additionally j is incremented by being multiplied by 2 each time instead of increased by 1. This behavior is what causes the line to be n log(n).

Connor Brown

3) Find the big-Θ for the following expressions:

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

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

The answer is A. This is because in big theta notation the answer must work for both big-O and big-Omega. B doesn't work because it is not proper format. C is incorrect because it only works for big-O and would not work for big-Omega. Then D doesn't work for big-O but does work for big-Omega. In the end the only one that woks for both is A.

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

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

The answer is D. This is due to the fact that as the integers are being added the total number of calculations would be (n(n-1)^2)/2 which when evaluated comes to be n^3. So therefore the answer would be big-Theta of 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

The answer is B. This is because all the program will have to do is iterate through each term and find a match which will just be (n-1) times. Simplifying we get n as the answer.

5) 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):

Connor Brown

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.

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) but we do not know the cost of this algorithm per unit. However, we know that 20 students are registered to CS251.

Approach C) Assume you are provided with an ($n\ log(n)$) sorting algorithm and this algorithm costs 50 time units per step to perform.

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

5-1) What should be the cost per time unit for Approach B so it takes the same amount of time as Approach A?

a. 1
b. 5
c. 100
d. 2000

The answer is B. This is because for Approach A it takes 100 units * the number of students which we know in Approach c is 20. Therefore for approach A the total is 2000 units. If we take this answer and divide by 20^2 (400) then we get 5 as the cost per unit.

5-2) How many students should register to CS251 so Approaches A and C take the same amount of time?

a. 20
b. 50
c. 100
d. 200

The answer is C. If we set the two equations equal to each other, 100n = 50n log(n) and solve for n we will get n = 100. Then we confirm the answer to be true by plugging in 100 for n in both equations and get 10,000 for each.

Submit Instructions:

Connor Brown

The homework must be turned in by the due date and time via Blackboard.