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. $O(log(n))$

nlog(n^2) is the highest order terms
it has more growth rate and can better present
the actual growth of the whole expression

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

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

2^n is the highest order terms
it has more growth rate and can better present
the actual growth of the whole expression

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

2-1)

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

operations

O(n)

n(n+1)/2 = O(n^2)

same as last one, now it becomes O(n^3)

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

2-2)

```
int sum = 0;
for(int i = 1; i <= n; i*=2)
  for(int j = 1; j <= n^2; j++)
    sum += sum;
```

Operations

n

n*n^2=n^3

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

2-3)

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

Operations

n

n(n+1)/2 = O(n^2)

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

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. $\Theta(n\ log(n))$

n is lower order term compared to 3n^2
we can ignore it

3-2) $\sum_{i=0}^{n} i^2$ = 0 + 1 + 2 + 4 + ... + n^2 = n(n+1)(2n+1)/6

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

so it is bounded by c1*n^3 and c2*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. $log(n)$ with binary search, worst-case performance is O(logn)
b. n
c. $n\,log(n)$
d. 1

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):

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 $O(n\,log(n))$ 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?

Approach A, n step, time= 100n
Approach B, n^2 step, time = cn^2 (c is time units per step)
Approach C, nlog(n) step, time = 50nlog(n)

a. 1
√ b. 5
c. 100
d. 2000

if n = 20, 100 * 20 = 2000 = c*20^2 = 400c
c =5

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

50nlog(n) = 100n

a. 20
b. 50
√ c. 100
d. 200

if n = 100
50*100*log(100) = 10000
100*100 = 10000
so 100 students

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