CSE 373: Section 2

Dictionaries and Algorithm Analysis

October 5th, 2017

Big O notation

For each of the following, show that $f(n) \in O(g(n))$

1. $f(n) = 3n$ $g(n) = 5n^2$

$5n^2 = 5n^2$

$3n^2 \leq 5n^2$ for $n \geq 1$

$3n \leq 5n^2$ for $n \geq 1$

Therefore, with $c = 1$ an $n_0 = 1$, $f(n) \in O(g(n))$

2. $f(n) = 17$ $g(n) = 32n + n * log(n)$

$32n + n\lg n = 32n + n\lg n$

$32n \leq 32n + n\lg n$ for $n \geq 1$

$32 \leq 32n + n\lg n$ for $n \geq 1$

$17 \leq 32n + n\lg n$

$\therefore$ $c = 1$ and $n_0 = 1$ $f(n) \in O(g(n))$

3. $f(n) = 121n^3$ $g(n) = 11n^3$

$121n^3 = 11 \cdot 11n^3$

$121n^3 \leq 11 \cdot 11n^3$ for all n.

Therefore $c = 11$ and $n_0 = 1$ and $f(n) \in G(n)$

Asymptotic Analysis

For the following methods, determine asymptotic runtime in terms of n

1.
```
void method1(int n, int sum){
    for(int i = 0; i<n*100;i++){
        for(int j = n; j > 0; j--){
            sum++;
        }
        for(int k = 0; k<i; k++){
            sum++;
        }
    }
}
```

$n \cdot 100n = n^2$

$\sum_{i=0}^{100n} i = \frac{(100n)(100n+1)}{2} = n^2$

$+$

$\boxed{O(n^2)}$

2.
```
void method2(int n, int sum){
    int j = n;
    while (j>2){
        sum++;
        j = j/2;
    }
}
```

n

$n/2$

$n/4$

$\vdots$

$\log_2 n$

$\boxed{O(\log n)}$

Comparing runtimes

Order the following functions from fastest to slowest in terms of asymptotic runtime. If there are multiple in the same bigO family, indicate this.

- $n * (n^2 * log(n) + n)$
- n^2
- $10000n^3$
- $2^n + 3$
- $n^{\frac{1}{2}} + n + 128$
- $n + log(n) + \frac{n}{5}$

$$\left.\begin{array}{l} n + \log(n) + n/5 \\ \sqrt{n} + n + 128 \end{array}\right\} \text{both } O(n)$$

n^2

$\cancel{n^2}$ $10000n^3$

$n \cdot (n^2 \cdot \log(n) + n)$

$2^n + 3$

Dictionaries

Provide pseudocode for inserting into a sorted array. What are the best and worst case runtimes for the method you proposed?

given element k

peform a binary search to find the correct location

shift all other elements over

best case: last element in list because there is no shifting only the binary search takes time $O(\log(n))$

worst case: first element all n elements must be shifted $O(n)$