BST 234 - Lab 2

Divy Kangeyan

1/31/18

Run time complexity

$$f \in O(g) : \exists\ c > 0,\ \exists n_0\ s.t.\ \forall\ n > n_0 : f(n) \leq c * g(n)$$

Intuitively this means f (n) is O(g(n)) if f (n) grows at most as fast as some constant times g(n) for large n.
This is read as ''f(x) is big-Oh of g(x)" or ''g asymptotically dominates f."

Properties

- $f = O(f)$
- $O(O(f)) = O(f)$
- $kO(f) = O(f)$
- $O(f + k)$
- $O(f) + O(g) = O(\max(f, g))$
- $O(f) * O(g) = O(f * g)$

Example 1

Show that:

$$n^2 \ is \ not O(n)$$

Solution: Suppose $\exists$ constants c and n_o for which:

$$n^2 \leq cn, \forall \ n > n_o$$

by *dividing* both sides of the inequality *by n*, then $n \leq c$ must hold $\forall \ n > n_o$. ※
A contradiction!

Example 2

Show that :

$$2^{n+1} = O(2^n)$$

Solution:

$$2^{n+1} \leq 2^n * 2 \ , \forall n \geq 0$$

$$\therefore \ 2^{n+1} = O(2^n)$$

Example 3

Show that:

$$2^{2n} \textit{ is not } O(2^n)$$

Solution: If the above was true,there would exist n_0 and c such that $n > n_0$:

$$2^n * 2^n = 2^{2n} \leq c * 2^n$$

, so $2^n \leq c$ for $n > n_0$ which is clearly impossible since c is a constant.

Run -Time Complexity - Alternate Definition

Assume g(n) $\neq$ 0 near ∞.

$$f \in O(g) \iff \limsup_{n\to\infty} \frac{f(n)}{g(n)} < \infty$$

$$\limsup_{n\to\infty} h(n) = \lim_{n\to\infty} \left(\sup_{m\geq n} h(m) \right)$$

(i.e. the limit of the least upper bound)

f in O(g)? some examples:

$$\lim_{n\to\infty} \frac{n}{2n} = \frac{1}{2},\ \ n \textit{ is } O(2n)$$

$$\lim_{n\to\infty} \frac{2n}{n} = 2,\ \ 2n \textit{ is } O(n)$$

$$\lim_{n\to\infty} \frac{n}{\sqrt{n}} = \infty,\ \ n \textit{ is not } O(\sqrt{n})$$

Exercise 1

Prove:

$$log(log(n)) = O(log^2 n)$$

Hint: L'Hospital's rule

Exercise 1

Prove:

$$log(log(n)) = O(log^2 n)$$

$$\lim_{n \to \infty} \frac{log(log(n))}{log^2 n} =$$

Exercise 1

Prove:

$$log(log(n)) = O(log^2 n)$$

$$\lim_{n\to\infty} \frac{log(log(n))}{log^2 n} = \lim_{n\to\infty} \frac{\frac{1}{nlog(n)}}{\frac{2log(n)}{n}} =$$

Exercise 1

Prove:

$$log(log(n)) = O(log^2 n)$$

$$\lim_{n\to\infty} \frac{log(log(n))}{log^2 n} = \lim_{n\to\infty} \frac{\frac{1}{nlog(n)}}{\frac{2log(n)}{n}} = \frac{1}{2log^2 n} = 0$$

$$\therefore log(log(n)) = O(log^2 n)$$

Exercise 2

Prove:

$$\lim_{n\to\infty} \frac{2^{n+1}-1}{2^n} \text{ is } O(1)$$

Exercise 2

Prove:

$$\lim_{n\to\infty} \frac{2^{n+1}-1}{2^n} \; is \; O(1)$$

Solution:

$$\lim_{n\to\infty} \frac{2^{n+1}-1}{2^n} = \lim_{n\to\infty} \frac{(n+1)2^n}{n2^{n-1}} = \lim_{n\to\infty} \frac{2n+2}{1} = 2 = O(1)$$

Exercise 3

Order from lowest to highest:

I. O(log_{10} n)
II. O(n!)
III. O(2^n)
IV. O(1)
V. O(log_2 n)
VI. O(nlog n)
VII. O(n^2)
VIII. O(n)

1. O(1)

Exercise 3

Order from lowest to highest:

I. O(log_{10} n)
II. O(n!)
III. O(2^n)
IV. O(1)
V. O(log_2 n)
VI. O(nlog n)
VII. O(n^2)
VIII. O(n)

1. O(1)
2. O(log_{10} n)=O(log_2 n)

Exercise 3

Order from lowest to highest:

I. O(log_{10} n)
II. O(n!)
III. O(2^n)
IV. O(1)
V. O(log_2 n)
VI. O(nlog n)
VII. O(n^2)
VIII. O(n)

1. O(1)
2. O(log_{10} n)=O(log_2 n)
3. O(n)

Exercise 3

Order from lowest to highest:

I. O(log_{10} n)
II. O(n!)
III. O(2^n)
IV. O(1)
V. O(log_2 n)
VI. O(nlog n)
VII. O(n^2)
VIII. O(n)

1. O(1)
2. O(log_{10} n)=O(log_2 n)
3. O(n)
4. O(nlog n)

Exercise 3

Order from lowest to highest:

I. O(log_{10} n)
II. O(n!)
III. O(2^n)
IV. O(1)
V. O(log_2 n)
VI. O(nlog n)
VII. O(n^2)
VIII. O(n)

1. O(1)
2. O(log_{10} n)=O(log_2 n)
3. O(n)
4. O(nlog n)
5. O(n^2)

Exercise 3

Order from lowest to highest:

I. $O(log_{10}\ n)$
II. $O(n!)$
III. $O(2^n)$
IV. $O(1)$
V. $O(log_2\ n)$
VI. $O(n\log\ n)$
VII. $O(n^2)$
VIII. $O(n)$

1. $O(1)$
2. $O(log_{10}\ n) = O(log_2\ n)$
3. $O(n)$
4. $O(n\log\ n)$
5. $O(n^2)$
6. $O(2^n)$

Exercise 3

Order from lowest to highest:

I. $O(log_{10}\ n)$
II. $O(n!)$
III. $O(2^n)$
IV. $O(1)$
V. $O(log_2\ n)$
VI. $O(n\log\ n)$
VII. $O(n^2)$
VIII. $O(n)$

1. $O(1)$
2. $O(log_{10}\ n) = O(log_2\ n)$
3. $O(n)$
4. $O(n\log\ n)$
5. $O(n^2)$
6. $O(2^n)$
7. $O(n!)$

The "=" sign in Big-Oh Notation

$$
\begin{aligned}
n &= O(n^4) \\
n^2 &= O(n^4) \\
n^3 &= O(n^4) \\
\implies n &= n^2 = n^3
\end{aligned}
$$

The "=" sign should be interpreted as a $\leq$.

The "$=$" sign in Big-Oh Notation

Is there an analog for $<$?

Little-Oh Notation

Assume g(n) $\neq$0 near ∞.

$$f \in O(g) \iff \limsup_{n \to \infty} \frac{f(n)}{g(n)} = 0$$

$$\limsup_{n \to \infty} h(n) = \lim_{n \to \infty} \left(\sup_{m \geq n} h(m) \right)$$

(i.e. the limit of the least upper bound)

What's the Difference?

True for **Big-Oh but** not Little-Oh:

- $x^3 \in O(x^3)$
- $x^3 \in O(x^3 + x)$
- $x^3 \in O(10x^3)$

True for **Little-Oh**:

- $x^2 \in O(x^3)$
- $x^3 \in O(x^4)$
- $x^3 \in O(x!)$

Exercise 4

Prove: $n^k \in O(b^n)$ and $b^n \notin O(n^k) \forall b > 1, n > 1, k \geq 0$

Hint: use the following theorem:
Let f(n), g(n) be two non-negative functions. Then:

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0 \implies f(n) \in O(g(n)) \text{ and } g(n) \notin O(f(n))$$

Use L'Hospitals!

Exercise 4

Prove: $n^k \in O(b^n)$ and $b^n \notin O(n^k) \forall b > 1, n > 1, k \geq 0$

Hint: use the following theorem:
Let f(n), g(n) be two non-negative functions. Then:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \implies f(n) \in O(g(n)) \text{ and } g(n) \notin O(f(n))$$

Use L'Hospitals!

$$\lim_{n\to\infty} \frac{n^k}{b^n} = \lim_{n\to\infty} \frac{kn^{k-1}}{nb^{n-1}} = \cdots \lim_{n\to\infty} \frac{k!}{b^n(logb)^k}$$

Insertion sort

1. Input an unsorted array A of length n
2. Iterate from $i=2:n$
3. Remove element i
4. Compare with sorted elements$<i$
5. Insert element i in sorted position

Exercise 4 - Implement the insertion sort in Python

```
def insertionSort(alist):
    for index in range(1,len(alist)):
            currentvalue = alist[index]
            position = index
            while position>0 and alist[position-1]>currentvalue:
                    alist[position]=alist[position-1]
                    position = position-1
                    alist[position]=currentvalue
    return(alist)
```

Exercise 5

What is the worst-case runtime complexity of the insertion sort?Prove it. Best case?

$$Hint: \sum_{n=1} i = \frac{n^2 + n}{2}$$

Exercise 5

What is the worst-case runtime complexity of the insertion sort? Prove it. Best case?

Solution: $O(n^2)$. Note that we execute the outer loop n times and in the worst case scenario, we must make i comparisons each iteration. Combine this with the hint on the previous slide. Best case is O(n).