CS-218 DATA STRUCTURES AND ALGORITHMS

LECTURE 6

BY UROOJ AINUDDIN

IN THE LAST LECTURE…

- We learned about the Big – O notation.
- We learned how to find Big – O notation of functions.
- We learned the rules of the Big – O notation.
- We learned how to find Big – O notation of code fragments.

ANALYSIS OF ALGORITHMS (ASYMPTOTIC ANALYSIS)

BOOK 1 CHAPTER 4

BOOK 2 CHAPTER 2

```
def findNeg(intList):
    n=len(intList)
    for i in range(n):
        if intList[i]<0:
            return i
    return None
```

What is the best case for this algorithm?
What is the worst case for this algorithm?

FIND THE BIG – O NOTATION FOR THE GIVEN CODE FRAGMENT

```
def findNeg(intList):
    n=len(intList)
    for i in range(n):
        if intList[i]<0:
            return i
    return None
```

There is a negative number at intList[0]. In the first iteration, we enter the body of the *if* and return 0.

$$O(1)$$

BEST CASE ANALYSIS

```
def findNeg(intList):
    n=len(intList)
    for i in range(n):
        if intList[i]<0:
            return i
    return None
```

a. There is a negative number at intList[n – 1]. The loop iterates n times and returns n – 1.
b. There isn't any negative number in intList. The loop iterates n times, and the function returns None.

$$O(n)$$

WORST CASE ANALYSIS

```
def findNeg(intList):
    n=len(intList)
    for i in range(n):
        if intList[i]<0:
            return i
    return None
```

Possible inputs:

a. There is a negative number at intList[0]. — p_0
b. There is a negative number at intList[1]. — p_1

$\vdots$

c. There is a negative number at intList[n – 1]. — p_{n-1}
d. There is no negative number in intList. — q

Probabilities: p_0, p_1, $\vdots$, p_{n-1}, q

$$T(n) = \sum_{i=0}^{n-1} (locations\ checked\ if\ neg.no.is\ at\ i)(prob.\ at\ i) + n(q)$$

$$T(n) = 1.p_0 + 2.p_1 + \cdots + n.p_{n-1} + n.q$$

AVERAGE CASE ANALYSIS

```
def findNeg(intList):
    n=len(intList)
    for i in range(n):
        if intList[i]<0:
            return i
    return None
```

$$T(n) = 1.p_0 + 2.p_1 + \cdots + n.p_{n-1} + n.q$$

Assume intList is very large, and the probability of not finding a negative number at any index is very small.

$$q \approx 0$$

Assume intList is not sorted in any order; then the probability of finding a negative number at any index is equal.

$$p_0 = p_1 = \cdots = p_{n-1} = p$$

We know that the sum of probabilities of all outcomes of an experiment is 1.

$$p_0 + p_1 + \cdots + p_{n-1} + q = 1$$

$$np = 1$$

$$p = \frac{1}{n}$$

AVERAGE CASE ANALYSIS

```
def findNeg(intList):
    n=len(intList)
    for i in range(n):
        if intList[i]<0:
            return i
    return None
```

$$T(n) = 1.p_0 + 2.p_1 + \cdots + n.p_{n-1} + n.q$$

$$p_0 = p_1 = \cdots = p_{n-1} = p = \frac{1}{n}$$

$$\mathrm{q} \approx 0$$

$$T(n) = 1.p + 2.p + \cdots + n.p$$

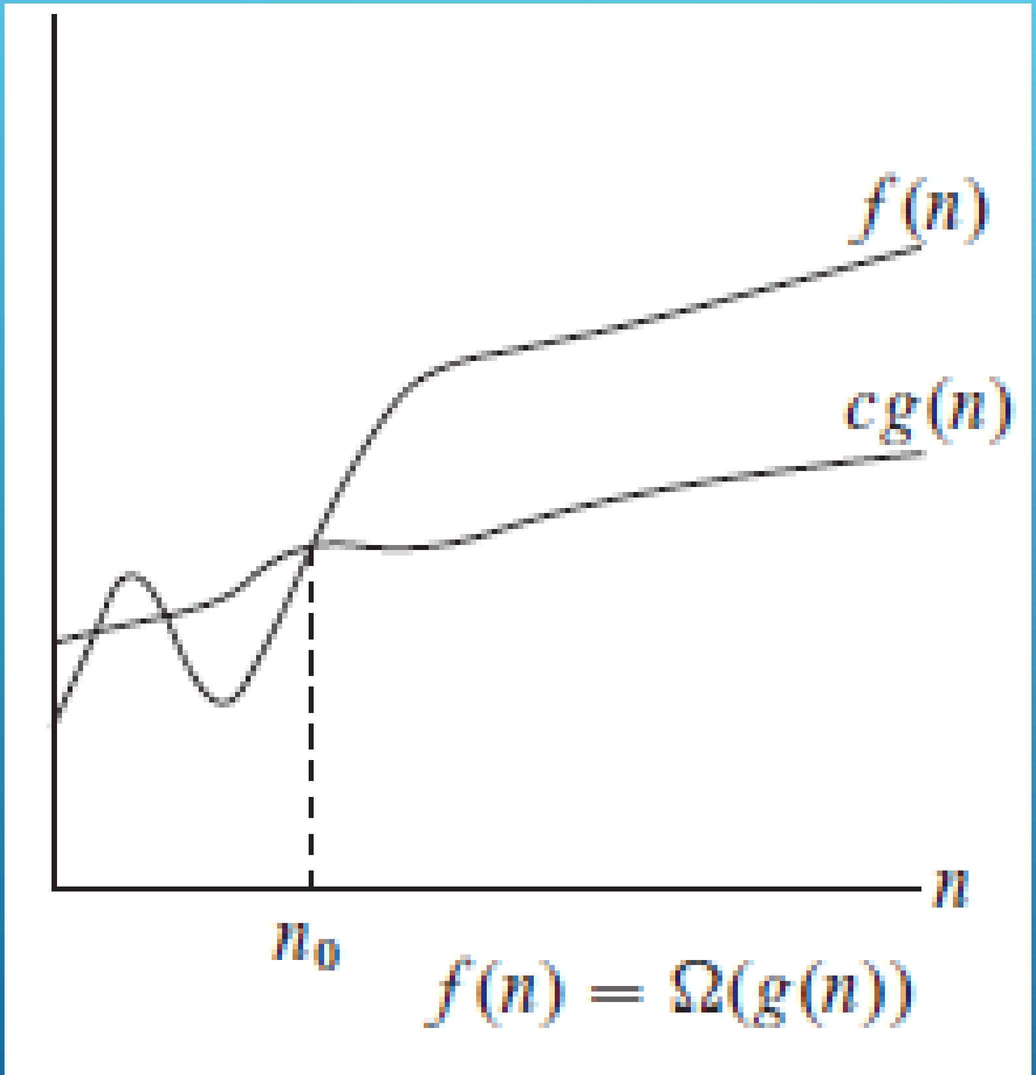
$$T(n) = (1 + 2 + \cdots + n)(\frac{1}{n})$$

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

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

$$\mathrm{O(n)}$$

AVERAGE CASE ANALYSIS



- Given functions $f(n)$ and $g(n)$, we say that $f(n)$ *is* $\Omega(g(n))$ if there are positive constants c and n_0 such that $f(n) \geq cg(n)$ for $n \geq n_0$.
- Ω notation provides a lower bound on running time of an algorithm.

BIG – OMEGA NOTATION

- $f(n) = 8n + 5$
- We need to prove $f(n) \geq cg(n)$
- We know that

$$8n + 5 \geq 8n$$

- This is true for all values of n, but we are interested in values of n greater than 0, so the smallest possible value of n, $n_0 = 1$.
- Take $c = 8$
- So $g(n) = n$
- $f(n)\ is\ \Omega(n)$
- The linear function is a lower bound on the growth of $f(n)$.

FIND $g(n)$ SUCH THAT $f(n)\ is\ \Omega\big(g(n)\big)$.

- $f(n) = 3n\log n - 2n$
- We need to prove $f(n) \geq cg(n)$
- We know that

$$3n\log n - 2n = n\log n + 2n\log n - 2n$$

$$3n\log n - 2n = n\log n + 2n(\log n - 1) \geq n\log n$$

$$2n(\log n - 1) \geq 0$$

$$n \geq 0 \qquad \log n - 1 \geq 0$$

$$\log n \geq 1$$

$$n \geq 2$$

FIND $g(n)$ SUCH THAT $f(n)$ is $\Omega(g(n))$.

- $f(n) = 3n\log n - 2n$
- We need to prove $f(n) \geq cg(n)$
- We know that

$$3n\log n - 2n \geq n\log n + 2n\log n - 2n$$
$$3n\log n - 2n = n\log n + 2n(\log n - 1) \geq n\log n\,, \forall n \geq 2$$
$$3n\log n - 2n \geq n\log n\,, \forall n \geq 2$$

- The smallest possible value of n, $n_0 = 2$.
- Take $c = 1$
- So $g(n) = n\log n$
- $f(n)\ is\ \Omega(n\log n)$
- The linearithmic function is a lower bound on the growth of $f(n)$.

FIND $g(n)$ SUCH THAT $f(n)\ is\ \Omega\big(g(n)\big)$.

- $f(n) = 5n^2 - 3n + 2$
- We need to prove $f(n) \geq cg(n)$
- We know that

$$5n^2 - 3n + 2 \geq 5n^2 - 3n$$

$$5n^2 - 3n + 2 \geq 2n^2 + 3n^2 - 3n$$

$$5n^2 - 3n + 2 \geq 2n^2 + 3n(n-1) \geq 2n^2$$

$$n \geq 0 \qquad n - 1 \geq 0$$

$$n \geq 1$$

FIND $g(n)$ SUCH THAT $f(n)\ is\ \Omega(g(n))$.

- $f(n) = 5n^2 - 3n + 2$
- We need to prove $f(n) \geq cg(n)$
- We know that

$$5n^2 - 3n + 2 \geq 5n^2 - 3n$$
$$5n^2 - 3n + 2 \geq 2n^2 + 3n^2 - 3n$$
$$5n^2 - 3n + 2 \geq 2n^2 + 3n(n-1) \geq 2n^2, \forall n \geq 1$$

- The smallest possible value of n, $n_0 = 1$.
- Take $c = 2$
- So $g(n) = n^2$
- $f(n)\ is\ \Omega(n^2)$
- The quadratic function is a lower bound on the growth of $f(n)$.

FIND $g(n)$ SUCH THAT $f(n)\ is\ \Omega\big(g(n)\big)$.

- $f(n) = 8\log n - 2$
- We need to prove $f(n) \geq cg(n)$
- We know that

$$8\log n - 2 \geq 6\log n + 2\log n - 2$$
$$8\log n - 2 \geq 6\log n + 2(\log n - 1) \geq 6\log n$$
$$\log n - 1 \geq 0$$
$$\log n \geq 1$$
$$n \geq 2$$

FIND $g(n)$ SUCH THAT $f(n)\ is\ \Omega\big(g(n)\big)$.

- $f(n) = 8\log n - 2$
- We need to prove $f(n) \geq cg(n)$
- We know that

$$8\log n - 2 \geq 6\log n + 2\log n - 2$$
$$8\log n - 2 \geq 6\log n + 2(\log n - 1) \geq 6\log n\,, \forall n \geq 2$$

- The smallest possible value of n, $n_0 = 2$.
- Take $c = 6$
- So $g(n) = \log n$
- $f(n)\ is\ \Omega(\log n)$
- The logarithmic function is a lower bound on the growth of $f(n)$.

FIND $g(n)$ SUCH THAT $f(n)\ is\ \Omega\big(g(n)\big)$.

POINT TO NOTE

- The Big – Omega notation of a function is its fastest growing term disregarding the constants.
- $f(n) = 3^n + 2n^2 + 3$
  - $f(n)$ is $\Omega(3^n)$
- $f(n)$ is $\Omega(g(n))$ if and only if $g(n)$ is $O(f(n))$
- The Big – Omega notation provides the **tightest** asymptotic lower bound of $f(n)$.

- $f(n) = 5n^3 + 4n$
- All polynomial functions of degree less than 3 have smaller rate of growth than $f(n)$.
- The linearithmic function, polylogarithmic function, log log n and constant functions have smaller rate of growth than $f(n)$.
- So all functions with a smaller growth rate can be the Big – Ω notation of $f(n)$.
- But we need the tightest lower bound, the smaller growth rate function that is closest to f(n).

$f(n)\ is\ \Omega(n^d), \forall d < 3$

$f(n)\ is\ \Omega(n \log n)$

$f(n)\ is\ \Omega(log^a n), \forall a \geq 1$

$f(n)\ is\ \Omega(\log \log n)$

$f(n)\ is\ \Omega(1)$

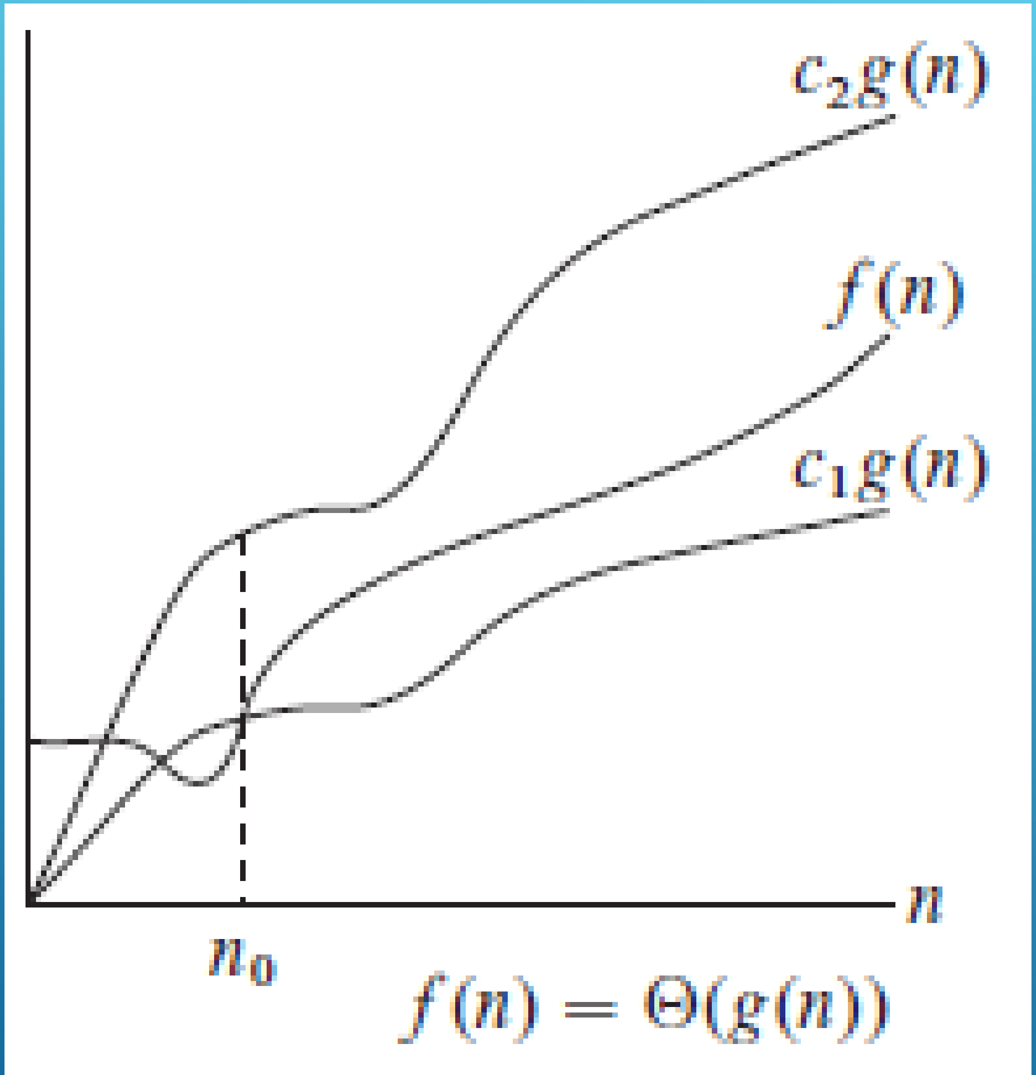
$f(n)\ is\ \Omega(n^3)$

$5n^3 + 4n \geq 5n^3$

WHICH FUNCTION WOULD BE THE MOST SUITABLE FOR BIG – Ω?

ASYMPTOTIC CONVENTIONS

- $5n^2 + 3n \geq 5n^2$ which means $5n^2 + 3n$ is $\Omega(5n^2)$ but it is poor taste to mention 5,
  - $5n^2 + 3n$ is $\Omega(n^2)$
- $5n^2 + 3n \geq 5n$ which means $5n^2 + 3n$ is $\Omega(n)$ but it is poor taste to use a lower order function for Big-Omega notation when a higher order function is available.
  - $5n^2 + 3n$ is $\Omega(n^2)$
- $7n^2 \geq n^2 + 6 \log n$ but it is poor taste to use two terms to describe Big-Omega notation and say $7n^2$ is $\Omega(n^2 + 6 \log n)$. Instead we use a single term that would suffice.
  - $7n^2$ is $\Omega(n^2)$



- Given functions $f(n)$ and $g(n)$, we say that $f(n)$ *is* $\Theta(g(n))$ if there are positive constants c_1, c_2 and n_0 such that

  $$c_1 g(n) \leq f(n) \leq c_2 g(n)$$

  for $n \geq n_0$.
- Θ notation provides a function that is both the upper and the lower bound on the running time of an algorithm.

BIG – THETA NOTATION

MORE ABOUT BIG - Θ

- For any two functions $f(n)$ and $g(n)$, $f(n)$ is $\Theta g(n)$ if and only if

  $$f(n) \text{is } O\big(g(n)\big)$$

  and

  $$f(n) \text{ is } \Omega\big(g(n)\big).$$

- Θ notation exists for all polynomials.

- $f(n) = 3n \log n + 4n + 5$
- We need to prove $c_1 g(n) \leq f(n) \leq c_2 g(n)$
- We know that

$$3n \log n \leq 3n \log n + 4n + 5 \leq 12n \log n$$

- This is true for all values of n greater than 1, so the smallest possible value of n, $n_0 = 2$.
- Take $c_1 = 3, c_2 = 12$
- So $g(n) = n \log n$
- $f(n)\ is\ \Theta(n \log n)$
- The linearithmic function grows with $f(n)$.

FIND $g(n)$ SUCH THAT $f(n)\ is\ \Theta\big(g(n)\big)$.

LITTLE – O AND LITTLE - OMEGA

- Given functions $f(n)$ and $g(n)$, we say that $f(n)\ is\ o(g(n))$ if there are positive constants c and n_0 such that $f(n) < cg(n)$ for $n \geq n_0$.
- Given functions $f(n)$ and $g(n)$, we say that $f(n)\ is\ \omega(g(n))$ if there are positive constants c and n_0 such that $f(n) > cg(n)$ for $n \geq n_0$.
- $f(n)\ is\ \omega(g(n))$ if and only if $g(n)\ is\ o(f(n))$

- $Let\ \lim_{n\to\infty}\frac{f(n)}{g(n)}\ be\ c.$
- $If\ c=0, f(n)\ is\ o\big(g(n)\big)$
- $If\ c=\infty, f(n)\ is\ \omega\big(g(n)\big)\ or\ g(n) is\ o(f(n))$
- $If\ c\in\mathbb{R}\ and\ c\neq 0, f(n)\ is\ \Theta(g(n))$

If you see an indeterminate form during calculation of limit, you must apply L'Hopital's Rule.

USING CALCULUS

- In a dare-devil game show, the contestant was asked to pick one of two boxes, and eat the insects inside it, to move to the next level.
- The boxes were marked A and B, but the contestant could not see the markings.
- Box A could hold 15-35 insects.
- Box B could hold 10-50 insects.
- Worst case for the contestant: Either he picks A and it has 35 insects, or he picks B and it has 50 insects.
- Best case for the contestant: Either he picks A and it has 15 insects, or he picks B and it has 10 insects.

A STORY…

Case	Lower bound	Upper bound
Best	10	15
Worst	35	50

1. Provide values of c (or c_1 and c_2) and n_0 to prove:
   a) $12n^3 + 5n^2 + 20n + 30$ is $\Theta(n^3)$
   b) $3n^2 + n \log n$ is $\Omega(n^2)$
   c) $(n \log n)^2 + 0.01n^2$ is $O(n^3)$

HOMEWORK

2. Using calculus, find the asymptotic relationship between these functions:

   a) $\mathrm{f(n)} = \mathrm{n}^{1.5} + \mathrm{n} \ln n\,, g(n) = \mathrm{n} \log \mathrm{n}$
   b) $\mathrm{f(n)} = (\mathrm{n}^3 + 4n + 10)^{100}, g(n) = \mathrm{n}^3$
   c) $\mathrm{f(n)} = 2^{2n}, g(n) = 2^n$
   d) $\mathrm{f(n)} = 2^{2n-1} - 1, g(n) = 2^{2n}$
   e) $\mathrm{f(n)} = \log_{10} n\,, g(n) = \log \mathrm{n}$

HOMEWORK

SO WHAT DID WE LEARN TODAY?

- We learned to analyze for best, worst and average case complexities.
- We learned about the remaining four asymptotic notations
- We realized that we could use calculus as a tool to find out the asymptotic relationship between two functions.

THINGS TO DO

Read

- the book!

Note

- your questions and put them up in the relevant online session.

Email

- suggestions on content or quality of this lecture at uroojain@neduet.edu.pk