SCC.121: ALGORITHMS AND COMPLEXITY
Big Ω and Θ notations

Emma Wilson, Ibrahim Aref

Today's Lecture

Aim: To introduce big Ω and big Θ notations

Learning objectives:

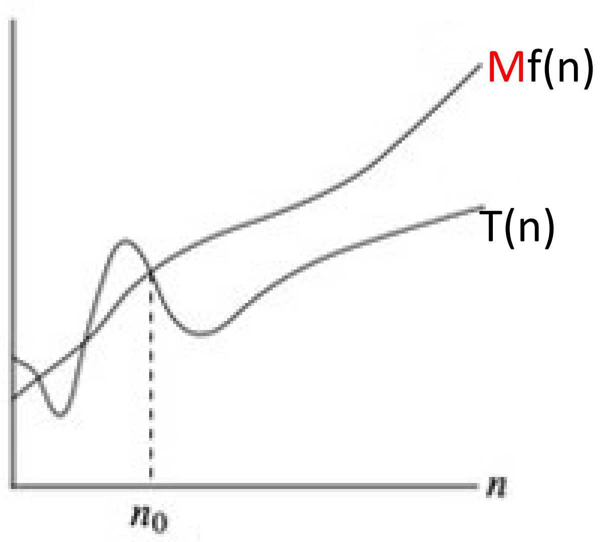
- To know how the growth of functions can be described using big Ω and big Θ notations and how these relate to big O
- To be able to define the growth of functions using big O, big Ω and big Θ

Outline

- **Asymptotic Growth**
  - **Big O notation**
  - **Big Ω notation**
  - **Big Θ notation**
- Linear Search in Big O, Big Ω and Big Θ
- Questions

The Big-O Notation

- **The formal definition of Big O:**
  - $T(n) \in O(f(n))$ if there are positive constants M and n_0 such that
    - $T(n) \leq M \times f(n)$ for all $n \geq n_0$



$T(n) \in O(f(n))$

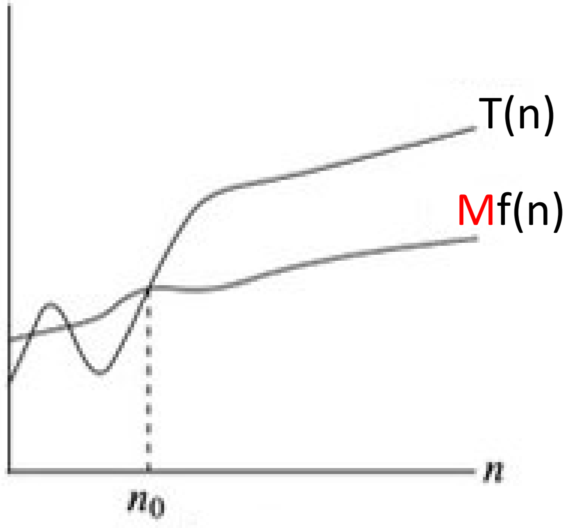
The Big Ω Notation

- **The formal definition of Big Ω:**
  - Let T(n) and f(n) be two positive functions from the integers or the real numbers to the real numbers
  - T(n) is Ω(f(n)) if even as n becomes arbitrarily large, T(n)'s growth is bounded from **below** by f(n), meaning it grows no slower than f(n)
  - $T(n) \in \Omega(f(n))$ if there are positive constants M and n_0 such that
    - $T(n) \geq M \times f(n)$ for all $n \geq n_0$

The Big Ω Notation

Lancaster University

- **The formal definition of Big Ω:**
  - $T(n) \in \Omega(f(n))$ if there are positive constants M and n_0 such that
    - $T(n) \geq M \times f(n)$ for all $n \geq n_0$



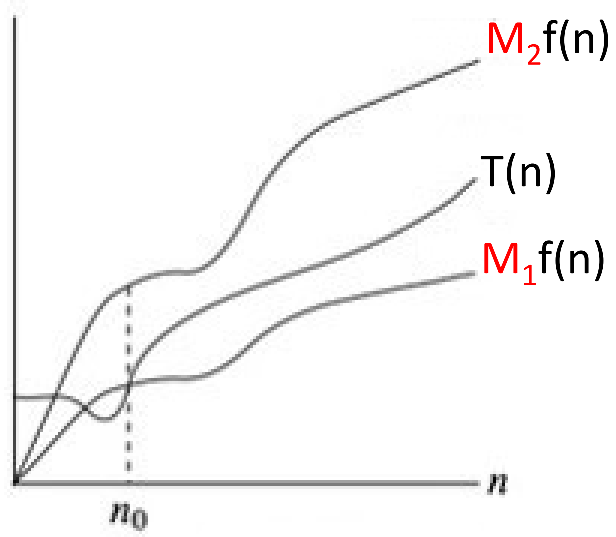
$T(n) \in \Omega(f(n))$

The Big Θ Notation

- **The formal definition of Big Θ:**
  - Let T(n) and f(n) be two positive functions from the integers or the real numbers to the real numbers
  - T(n) is Θ(f(n)) if even as n becomes arbitrarily large, T(n)'s growth is bounded from **above and below** by f(n), meaning it grows no faster and no slower than f(n)
  - $T(n) \in \Theta(f(n))$ if there are positive constants M_1, M_2 and n_0 such that
    - $M_1 \times f(n) \leq T(n) \leq M_2 \times f(n)$ for all $n \geq n_0$
  - It means $T(n) \in \Theta(f(n))$ **if** $T(n) \in O(f(n))$ **AND** $T(n) \in \Omega(f(n))$

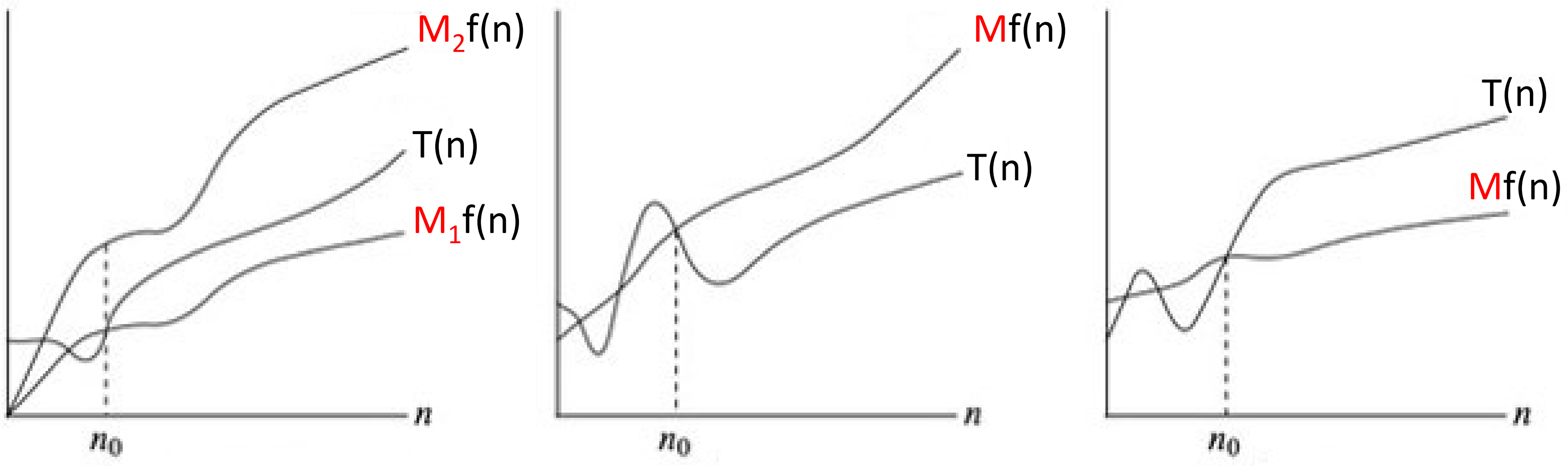
The Big Θ Notation

- **The formal definition of Big Θ:**
  - $T(n) \in \Theta(f(n))$ if there are positive constants M_1, M_2 and n_0 such that
    - $M_1 \times f(n) \leq T(n) \leq M_2 \times f(n)$ for all $n \geq n_0$



$T(n) \in \Theta(f(n))$

Big O, Big Ω, Big Θ



$T(n) \in \Theta(f(n))$ $T(n) \in O(f(n))$ $T(n) \in \Omega(f(n))$

Example#1

- **Example#1:**
  - $T(n) = 3n + 4$
  - $f(n) = n$
  - Show that $T(n)$ is $\Theta(f(n))$ which means $T(n)$ is $\Theta(n)$
- **Solution:**
  - We need to prove that $T(n) \in O(n)$ **AND** $T(n) \in \Omega(n)$

Example#1

- **Example#1:**
  - $T(n) = 3n + 4$
  - $f(n) = n$
  - Show that $T(n)$ is $\Theta(f(n))$ which means $T(n)$ is $\Theta(n)$
- **Solution:**
  - We have proved that $T(n) \in O(n)$ (Week 14 Lecture 2)
  - For $n \geq 1$ we have: $T(n) = 3n + 4 \leq 3n + 4n$
  - So, $T(n) = 3n + 4 \leq 7n$
  - Therefore, for $M=7$ and $n_0=1$ ➔ $T(n) \leq 7n$ for all $n \geq 1$
  - $T(n) \in O(n)$

Example#1

- **Example#1:**
  - $T(n) = 3n + 4$
  - $f(n) = n$
  - Show that T(n) is $\Theta(f(n))$ which means T(n) is $\Theta(n)$
- **Solution:**
  - We have proved that $T(n) \in O(n)$ (Week 14 Lecture 2)
  - Now, we only need to show that $T(n) \in \Omega(n)$
  - For $n \geq 0$ we have: $T(n) = 3n + 4 \geq 2n + 0$

  - So, $T(n) = 3n + 4 \geq 2n$

  - Therefore, for $M=2$ and $n_0=0$ ➔ $T(n) \geq 2n$ for all $n \geq 0$
  - $T(n) \in \Omega(n)$

  - Since $T(n) \in O(n)$ **AND** $T(n) \in \Omega(n)$, we can say $T(n) \in \Theta(n)$

In General

- **Examples:**
  - T(n) = $C_1 \times N + C_0$ → O(N) Ω(N) Θ(N)
  - T(n) = $C_2 \times N^2 + C_1 \times N + C_0$ → $O(N^2)$ $\Omega(N^2)$ $\Theta(N^2)$
  - T(n) = $C_3 \times N^3 + C_2 \times N^2 + C_1 \times N + C_0$ → $O(N^3)$ $\Omega(N^3)$ $\Theta(N^3)$
  - T(n) = $C_k \times N^k + C_{k-1} \times N^{k-1} + \cdots + C_1 \times N + C_0$ → $O(N^k)$ $\Omega(N^k)$ $\Theta(N^k)$
- More examples:
  - T(n) = $C_2 \times N + C_1 \log N + C_0$ → O(N) Ω(N) Θ(N)
  - T(n) = $C_2 \times N^k + C_1 2^N + C_0$ → $O(2^N)$ $\Omega(2^N)$ $\Theta(2^N)$

Outline

- Asymptotic Growth
  - Big O notation
  - Big Ω notation
  - Big Θ notation
- **Linear Search in Big O, Big Ω and Big Θ**
- Questions

Linear Search

```
int isInArray(int theArray[], int N, int iSearch)
{

        for (int i = 0; i < N; i++)
                if (theArray[i] == iSearch)
                        return 1;

        return 0;
}
```

Linear Search
Big O, Big Ω and Big Θ

- What is the growth rate in the Big O, Big Ω and Big Θ notation?
  - **Best Case:** T(N) = 4
    - **T(N) = Constant** → $O(1)$ Ω(1) Θ(1)
  - **Worst Case:** T(N) = 3N+3
    - **T(N) = C_1 x N + C_2** → $O(N)$ Ω(N) Θ(N)
    - **C_1 and C_2 are constant**
  - **Average case:** T(N) = $\left(\frac{3}{2}P + 3 - 3\mathrm{P}\right)\mathrm{N} + \left(\frac{5}{2}P + 3 - 3P\right)$
    - **T(N) = C_1 x N + C_2** → $O(N)$ Ω(N) Θ(N)
    - **C_1 and C_2 are constant**

Linear Search
Big O, Big Ω and Big Θ

Case	T(n)	Θ
Worst	linear function of n	$\Theta(n)$, $O(n)$, $O(n^2)$, $\Omega(n)$, $\Omega(logn)$
Average	linear function of n	$\Theta(n)$ but $O(n)$, $O(n^2)$, $\Omega(n)$, $\Omega(logn)$
Best	constant	$\Theta(1)$ but $O(n)$?, $O(n^2)$, $O(logn)$

When No Case is Mentioned

- Which are **TRUE**?
  - Linear search is $O(n)$
  - Linear search is $\Omega(1)$
  - Linear search is $\Theta(n)$

Linear search is O(n)?

- Linear search is O(n)
  - **TRUE**: There is no input of size n for which the runtime is not bounded from above by n
  - Best case $O(1)$, $O(\log n)$, $O(n)$, ….
  - Average case $O(n)$, $O(n \log n)$, $O(n^2)$, …..
  - Worst case $O(n)$, $O(n \log n)$, $O(n^2)$, …..

Linear search is $\Omega(1)$?

- Linear search is $\Omega(1)$
  - **TRUE:** There is no input of size n that takes less than a constant amount of time
  - **True** for every algorithm
  - Best case $\Omega(1)$
  - Average case $\Omega(n)$, $\Omega(\log n)$, $\Omega(1)$
  - Worst case $\Omega(n)$, $\Omega(\log n)$, $\Omega(1)$

Linear search is Θ(n)?

- Linear search is Θ(n)
  - **FALSE:** There is an input of size n, the best case, specifically, for which the runtime is Θ(1)
  - Best case $\Theta(1)$
  - Average case $\Theta(n)$
  - Worst case $\Theta(n)$

Linear Search
Big O, Big Ω and Big Θ

Case	T(n)	Θ
Worst	linear function of n	$\Theta(n)$, $O(n)$, $\boxed{O(n^2)}$, $\Omega(n)$, $\boxed{\Omega(logn)}$
Average	linear function of n	$\Theta(n)$ but $O(n)$, $O(n^2)$, $\Omega(n)$, $\Omega(logn)$
Best	constant	$\Theta(1)$ but $\boxed{O(n)?}$, $\boxed{O(n^2)}$, $\boxed{O(logn)}$

Outline

- Asymptotic Growth
  - Big O notation
  - Big Ω notation
  - Big Θ notation
- Linear Search in Big O, Big Ω and Big Θ
- **Questions**

Question Q1:

State which of the following are true about linear search.

ii. $\Theta(n)$ in the worst case

iii. $O(n)$

iv. $\Omega(n)$

v. $\Omega(n)$ in the worst case

vi. $O(n)$ in the worst case

vii. $\Omega(n^2)$ in the worst case

viii. $O(n^2)$

ix. $\Omega(\log n)$

slido

State which of the following are true about linear search (you can select multiple options)

ⓘ Start presenting to display the poll results on this slide.

Answer Q1:

- i. $\Theta(n)$ **false** , because it takes $\Theta(1)$ in the best case
- ii. $\Theta(n)$ in the worst case **true**
- iii. $O(n)$ **true**: because there is no input for which it takes longer
- iv. $\Omega(n)$ **false**: because it takes $\Omega(1)$ in the best case
- v. $\Omega(n)$ in the worst case **true**: because it takes $\Theta(n)$ in the worst case
- vi. $O(n)$ in the worst case **true**: because it takes $\Theta(n)$ in the worst case
- vii. $\Omega(n^2)$ in the worst case **false**: because it takes Ω (n) in the worst case
- viii. $O(n^2)$ **true**: because it takes $O(n)$
- ix. $\Omega(\log n)$ **false** because it takes $\Omega(1)$ in the best case

Questions: Q2

Given a time complexity $T(n) = 0.01\, n^2 + 4\log(n) + 3$

Based on the definitions of big-O, big-Omega and big-Theta, which of the following statements is **false**?

i. $T(n) \in \Theta(n^2)$

ii. $T(n) \in \Omega(\log n)$

iii. $T(n) \in O(\log n)$

iv. $T(n) \in \mathrm{O}(n!)$

Given a time complexity $T(n)$=0.01 n^2+4 log(n)+3

Based on the definitions of big-O, big-Omega and big-Theta, which of the following statements is false?

ⓘ Start presenting to display the poll results on this slide.

Answer: Q2

Given a time complexity $T(n) = 0.01\, n^2 + 4\log(n) + 3$

Based on the definitions of big-O, big-Omega and big-Theta, which of the following statements is **false**?

i. $T(n) \in \Theta(n^2)$

ii. $T(n) \in \Omega(\log n)$

iii. $T(n) \in O(\log n)$

iv. $T(n) \in \mathrm{O}(n!)$

iii. is the correct answer. T(n) is not $O(\log n)$ as big-O defines an upper bound, so T(n) is only big-O of functions that grow as fast (or faster) than the dominant term n^2

Question: Q3

Consider the code fragment below. Assume the inputs are two arrays of size N and M respectively. Determine the time complexity in the big- θ notation.

```
for (i = 0; i < N; i++) {
        Sequence of statements with θ(N) complexity
}
for (j = 0 ; j < M; j++) {
        Sequence of statements with θ(1) complexity
}
```

i) $\theta(\max(N,M))$
ii) $\theta(\max(N^2,M))$
iii) $\theta(N^2M)$
iv) $\theta(NM+N)$

slido

Consider the code fragment. Assume the inputs are two arrays of size N and M respectively. Determine the time complexity in the big- θ notation.

ⓘ Start presenting to display the poll results on this slide.

Answer: Q3

Consider the code fragment below. Assume the inputs are two arrays of size N and M respectively. Determine the time complexity in the big- θ notation.

```
for (i = 0; i < N; i++) {
       Sequence of statements with θ(N) complexity
}
for (j = 0 ; j < M; j++) {
       Sequence of statements with θ(1) complexity
}
```

i) $\theta(\max(N,M))$
ii) $\theta(\max(N^2,M))$
iii) $\theta(N^2M)$
iv) $\theta(NM+N)$

Two sequential loops – most dominant is the maximum of the time complexity of each loop.

- Loop 1: Runs N times and each iteration does θ(N)operations. So, loop 1 complexity: $\theta(N^2)$
- Loop 2: Runs M times and each iteration does θ(1) operations. So, loop 2 complexity θ(M)
- Overall: max of two loops, so **$\theta(\max(N^2,M))$**

Summary: Big O, Big Ω and Big-Θ

Big O	Big Ω	Big-Θ
It is like (<=) rate of growth of an algorithm is less than or equal to a specific value.	It is like (>=) rate of growth is greater than or equal to a specified value.	It is like (==) meaning the rate of growth is equal to a specified value.
The upper bound of algorithm is represented by Big O notation. Only the above function is bounded by Big O. Asymptotic upper bound is given by Big O notation.	The algorithm's lower bound is represented by Omega notation. The asymptotic lower bound is given by Omega notation.	The bounding of function from above and below is represented by theta notation. The exact asymptotic behaviour is done by this theta notation.
Big oh (O) – Upper Bound	Big Omega (Ω) – Lower Bound	Big Theta (Θ) – Tight Bound
It is defined as an upper bound	It is define a a lower bound and lower	It is define as tightest bound and tightest bound is the best of all the worst case times that the algorithm can take.
Mathematically: $T(n) \in O(f(n))$ if there are positive constants M and n_0 such that $T(n) \leq M \times f(n)$ for all $n \geq n_0$	Mathematically: $T(n) \in \Omega(f(n))$ if there are positive constants M and n_0 such that $T(n) \geq M \times f(n)$ for all $n \geq n_0$	Mathematically: $T(n) \in \Theta(f(n))$ if there are positive constants M1, M2 and n_0 such that $M1 \times f(n) \leq T(n) \leq M2 \times f(n)$ for all $n \geq n_0$

Summary

Today's lecture: looked at some examples of finding Big O, big Ω and big Θ

- T(n) is O(f(n)) if even as n becomes arbitrarily large, T(n)'s growth is bounded from **above** by f(n), meaning it grows no faster than f(n)
- T(n) is Ω(f(n)) if even as n becomes arbitrarily large, T(n)'s growth is bounded from **below** by f(n), meaning it grows no slower than f(n)
- T(n) is Θ(f(n)) if even as n becomes arbitrarily large, T(n)'s growth is bounded from **above and below** by f(n), meaning it grows no faster and no slower than f(n)
- The growth of functions is usually described using the big Θ notation to avoid confusion

Next Lecture: Abstract Data Types