# COMP 3711 Design and Analysis of Algorithms

## Tutorial 1: Asymptotic Analysis

# Asymptotic Notation: Quick Revision

**Upper bounds.** $T(n) = O(f(n))$
if exist constants $c > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$, $T(n) \leq c \cdot f(n)$.

**Lower bounds.** $T(n) = \Omega(f(n))$
if exist constants $c > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$, $T(n) \geq c \cdot f(n)$.

**Tight bounds.** $T(n) = \Theta(f(n))$
if $T(n) = O(f(n))$ and $T(n) = \Omega(f(n))$.

Note: Here "=" means "is", not equal.
More mathematically correct expression should be $T(n) \in O(f(n))$.

constant < logarithmic < polynomial < exponential
$9999^{9999^{9999}}$ < $\log^{10} n$ < $n^{0.1}$ < $n \log n < n^2$ < $2^n$

# Some Basic Properties

a) If $f(n) = O(g(n))$ and $g(n) = O(h(n)) \Rightarrow f(n) = O(h(n))$

b) If $f(n) = \Omega(g(n))$ and $g(n) = \Omega(h(n)) \Rightarrow f(n) = \Omega(h(n))$

c) If $f(n) = \Theta(g(n))$ and $g(n) = \Theta(h(n)) \Rightarrow f(n) = \Theta(h(n))$

d) If $f(n) = O(h(n))$ and $g(n) = O(h(n)) \Rightarrow f(n) + g(n) = O(h(n))$

e) If $f(n) = \Omega(h(n))$ and $g(n) = \Omega(h(n)) \Rightarrow f(n) + g(n) = \Omega(h(n))$

f) If $f(n) = \Theta(h(n))$ and $g(n) = \Theta(h(n)) \Rightarrow f(n) + g(n) = \Theta(h(n))$

Prove by definition

# Question 1

For each of the following statements, answer whether the statement is true or false.

a) $1000n^2 + 1000n = O(n^3)$ True

b) $n^2 - n = \Theta(n)$ False

c) $nlog(n) = O(n^2)$ True

d) $nlog(n) = \Theta(n^2)$ False

e) $\frac{n}{100} = \Omega(n)$ True

f) $12n + 2^n + n^3 = O(n^3)$ False

g) $2nlog(n) + n = \Theta(nlogn)$ True

# Question 2

Suppose $T_1(n) = O\big(f(n)\big)$ and $T_2(n) = O\big(f(n)\big)$. Which of the following are true? Justify your answers.

(a) $T_1(n) + T_2(n) = O\big(f(n)\big)$

(b) $\frac{T_1(n)}{T_2(n)} = O(1)$

(c) $T_1(n) = O\big(T_2(n)\big)$

# Question 2

Suppose $T_1(n) = O\big(f(n)\big)$ and $T_2(n) = O\big(f(n)\big)$.

Is the following true?

(a) $T_1(n) + T_2(n) = O\big(f(n)\big)$? True.

This was just basic property (d).

---

Example

$T_1(n) = 2n^4 + 3n^3$ and $T_2(n) = 5n^4 + 2n^2$

$T_1(n) = \mathrm{O}(n^4)$ and $T_2(n) = \mathrm{O}(n^4)$

$\Rightarrow\ T_1(n) + T_2(n) = \mathrm{O}(n^4)$

# Question 2

Suppose $T_1(n) = O\big(f(n)\big)$ and $T_2(n) = O\big(f(n)\big)$.

Is the following true?

(b) $\frac{T_1(n)}{T_2(n)} = O(1)$? False.

Counterexample: Set $T_1(n) = n^2, T_2(n) = n, \ f(n) = n^2$.

$\Rightarrow \ T_1(n) = O\big(f(n)\big), \ T_2(n) = O\big(f(n)\big)$

but $\quad \frac{T_1(n)}{T_2(n)} = n \neq O(1)$

(c) $T_1(n) = O\big(T_2(n)\big)$? False.

Use the same counterexample as in part (b).

$n^2 \neq \ \mathrm{O}(n)$

# Question 3

Let $f(n)$ be a function.

Suppose that, for all $i > 0$, $T_i(n) = O\big(f(n)\big)$.

Define $g_{k(n)} = \sum_{i=1}^{k} T_i(n)$

(a) For fixed $k$, is $g_k(n) = O\big(f(n)\big)$?

(b) Define $g(n) = g_n(n)$.

Is $g(n) = O\big(f(n)\big)$?

Is $g(n) = O\big(nf(n)\big)$?

# Question 3: (a)

(a) For fixed $k$, is $g_k(n) = O\big(f(n)\big)$? **Yes.**

Recall $g_k(n) = \sum_{i=1}^{k} T_i(n)$ where, for each $i$, $T_i(n) = O\big(f(n)\big)$.

We know (basic property (d)) that

if $U(n) = O\big(f(n)\big)$ and $V(n) = O\big(f(n)\big)$ then $(U(n) + V(n)) = O(f(n))$

(a) For fixed $k$, is $g_k(n) = O\big(f(n)\big)$?

Then $g_2(n) = T_1(n) + T_2(n) = O\big(f(n)\big)$

Iterating, using induction, shows that, for FIXED k

$$g_k(n) = \underbrace{g_{k-1}(n)}_{U(n)} + \underbrace{T_k(n)}_{V(n)} = O(f(n))$$

# Question 3:

(b) $g_k(n) = \sum_{i=1}^{k} T_i(n)$ where, for each $i$, $T_i(n) = O\big(f(n)\big)$.

$$\text{g}(n) = g_n(n) = \sum_{i=1}^{n} T_i(n)$$

---

Even though we just saw that, for FIXED $k$, $g_k(n) = O\big(f(n)\big)$,

It is NOT true that $g(n) = O\big(f(n)\big)$
or even that $g(n) = O\big(nf(n)\big)$.

We display a counterexample.

# A Counterexample

Set $T_{\text{i}}(n) = i \cdot n$, and $f(n) = n$.

=> $T_i(n) = O(f(n))$ for all FIXED $i \geq 1$.

=> $g_k(n) = \sum_{i=1}^{k} T_i(n) = \sum_{i=1}^{k} i \cdot n = n\frac{k(k+1)}{2}$

---

=> $g_k(n) = c_k n$ where $c_k = \frac{k(k+1)}{2}$

=> So, *for fixed* $k$, $g_k(n) = O(n)$

Which we also know is true from part (a)

---

But $\text{g}(n) = g_n(n) = n\frac{n(n+1)}{2} = \Theta(n^3)$

In particular,

$g(n)$ is NOT $O(f(n))$ or even $O\big(n\, f(n)\big)$!

$O(n)$ $O(n^2)$

# A Deeper Dive

Suppose that $T_i(n) = O\big(f(n)\big)$ for all $i \geq 0$ and set $g(n) = \sum_{i=1}^{n} T_i(n)$.

$$(*) \quad g(n) \overset{(a)}{=} \sum_{i=1}^{n} O(f(n)) \overset{(b)}{=} O\left(\sum_{i=1}^{n} f(n)\right) = O(n\, f(n))$$

IS NOT CORRECT. Why is this "proof" wrong?

---

Use the counterexample from previous page to unpack problem.

Set $T_{\mathrm{i}}(n) = i \cdot n$, and $f(n) = n$. $\quad g(n) = \Theta(n^3) \neq O\big(n\, f(n)\big) = O(n^2)$

The problem is that equalities (a) and (b) aren't **mathematically well defined**.
$O(\ )$ notation has a multiplicative constant associated with it, i.e.,
writing $T_i(n) = O(n)$ implies there is a constant $c_i$, s.t. $T_i(n) \leq c_i n$

The way that (a) and (b) are written imply that all of the constants are the same.
But they are NOT. $c_i = i$, so the constants are increasing.

# Question

Let $a_1, a_2, \ldots, a_n$ be a sequence that has the following property:

There exists some $k$ such that

$\forall i: 1 \le i < k,\ \ a_i > a_{i+1};$ $\forall i: k \le i < n,\ \ a_i < a_{i+1}$

Such a sequence is called **unimodal** with the unique minimum $a_k$.

*(The sequence goes down and then goes up, with minimum at $a_k$.)*

**Design an $O(\log n)$ algorithm for finding $k$ in a unimodal sequence $(n \geq 3)$.**

# Example

Example: $A = [10, \; 8, \; 6, \; \mathbf{5}, \; 25, \; 30, \; 40, \; 70, \; 90, \; 100]$

$A$ is unimodal with minimum $a_4 = \mathbf{5}$.

# Solution

Example: $A = [\,10,\ 8,\ 6,\ \mathbf{5},\ 25,\ 30,\ 40,\ 70,\ 90,\ 100]$

- We can binary search for the transition point where the items stop decreasing and start increasing.

- Define new array $B = [1, n-1]$ where

$$B[i] = \begin{cases} + & if\ \ A[i] > A[i+1] \\ - & if\ \ A[i] < A[i+1] \end{cases}$$

  For example $A$: $B = [\ +, +\ \ +,\ \ -,\ \ -,\ \ -,\ \ -\ ,\ \ -,\ \ -]$

- Unimodality implies that $B$ is in form $B = [+,\ +,\ +,\ \ldots, +,\ -, -, \ldots, -]$, where $\boldsymbol{k}$, the location of the minimum value in $A$, is the location of the first "$-$" in $B$.

- This $\boldsymbol{k}$ can then be found using an $O(\log n)$ binary search for the first "$-$" in $B[]$.

# Solution

Example: $A = \ [\, 10,\ \ 8,\ \ 6,\ \ \mathbf{5},\ \ 25,\ \ 30,\ \ 40,\ \ 70,\ \ 90,\ \ 100]$

- We can binary search for the transition point where the items stop decreasing and start increasing.

- Define new array $B = [1, n-1]$ where

$$B[i] = \begin{cases} + & if\ \ A[i] > A[i+1] \\ - & if\ \ A[i] < A[i+1] \end{cases}$$

For example $A$: $B = [\ +, +\ \ +,\ \ -,\ \ -,\ \ -,\ \ -,\ \ -,\ \ -]$

$\boldsymbol{k}$ can be found by an $O(\log n\,)$ binary search for the first "$-$" in $B[\,]$.

- No need to actually build $B[\,]$; $B$'s entries can be calculated in $O(1)$ time from $A[\,]$, so we may assume $B$ is given.

# Solution: More Details

$BSearch(i, j)$ will be the algorithm.
The call assumes $i < j$ and the invariant that $B[i] =+,\ B[j] =-$ .

The first call will be $BSearch(1, n-1)$, which satisfies this invariant.

$BSearch(i, j)$ will return the smallest $k \in [i \ldots j]$ such that B$[k] =-$ .

Example: $B = [\ +, +\ \ +,\ \ -,\ \ -,\ \ -,\ \ -\ \ ,\ \ -,\ \ -]$

**$BSearch(i, j)$**
If $(j = i + 1)$
  return$(j)$
Else
  $m = \left\lfloor \frac{i+j}{2} \right\rfloor$
  If $B[m] =+$
    $BSearch(m, j)$
  Else /* $B[m] =-$ */
    $BSearch(i, m)$

or

**$BSearch(i, j)$**
If $(j = i + 1)$
  return$(j)$
Else
  $m = \left\lfloor \frac{i+j}{2} \right\rfloor$
  If $A[m] > A[m+1]$
    $BSearch(m, j)$
  Else /* $A[m] < A[m+1]$ */
    $BSearch(i, m)$