Asymptotic Notation,
Review of Functions & Summations

Module – 1: Forenoon
Day – 1: 19-10-2020

19 October 2020

CSE 270 Design and Analysis of Algorithm,
Prof. Jayanthi. G, SRET,

0

Asymptotic Complexity

- Running time of an algorithm as a function of input size n **for large n**.

- Expressed using only the **highest-order term** in the expression for the exact running time.

  - Instead of exact running time, say $\Theta(n^2)$. **Theta**

- Describes behavior of function in the limit.

- Written using *Asymptotic Notation*.

CSE 270 Design and Analysis of Algorithm,
Prof. Jayanthi. G, SRET,

Asymptotic Notation

- **Theta Θ, Big O, Big Ω, small o, small Ω**

- Defined for functions over the natural numbers.

  - **Ex:** $f(n) = \Theta(n^2)$.

  - Describes how $f(n)$ grows in comparison to n^2.

- Define a ***set*** of functions; in practice used to compare two function sizes.

- The notations describe different rate-of-growth relations between the defining function and the defined set of functions.

Θ-notation

For function $g(n)$, we define $\Theta(g(n))$, big-Theta of n, as the set:

$\Theta(g(n)) = \{f(n) :$
$\exists$ **positive constants** c_1, c_2, **and** n_0, **such that** $\forall n \geq n_0$,
we have $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$
$\}$

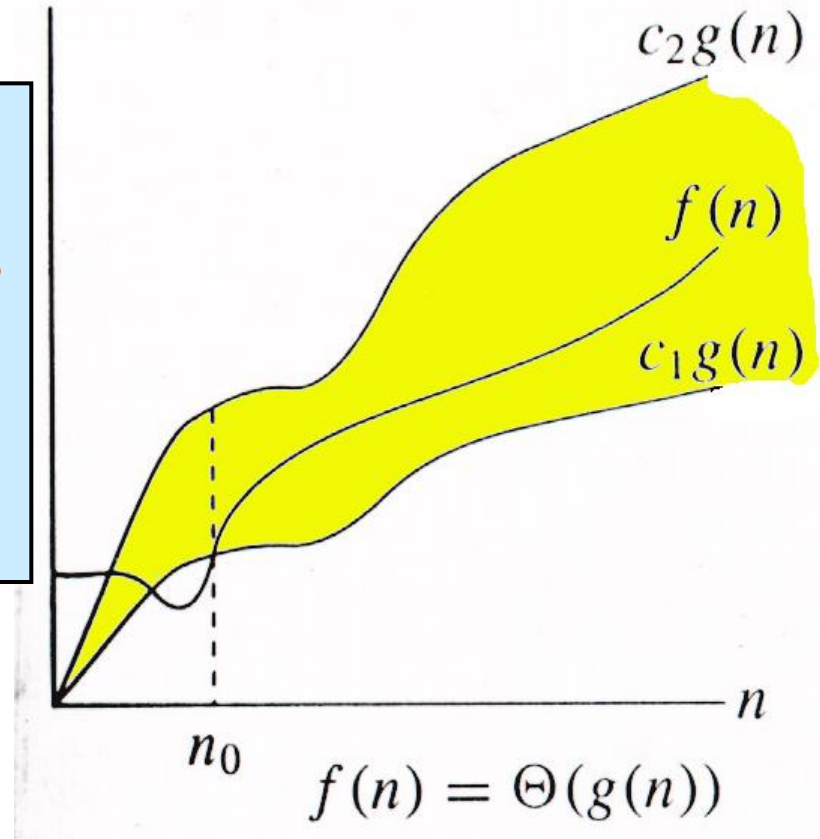*Intuitively*: Set of all functions that have the same *rate of growth* as $g(n)$.

$c_2 g(n)$

$f(n)$

$c_1 g(n)$

n

n_0
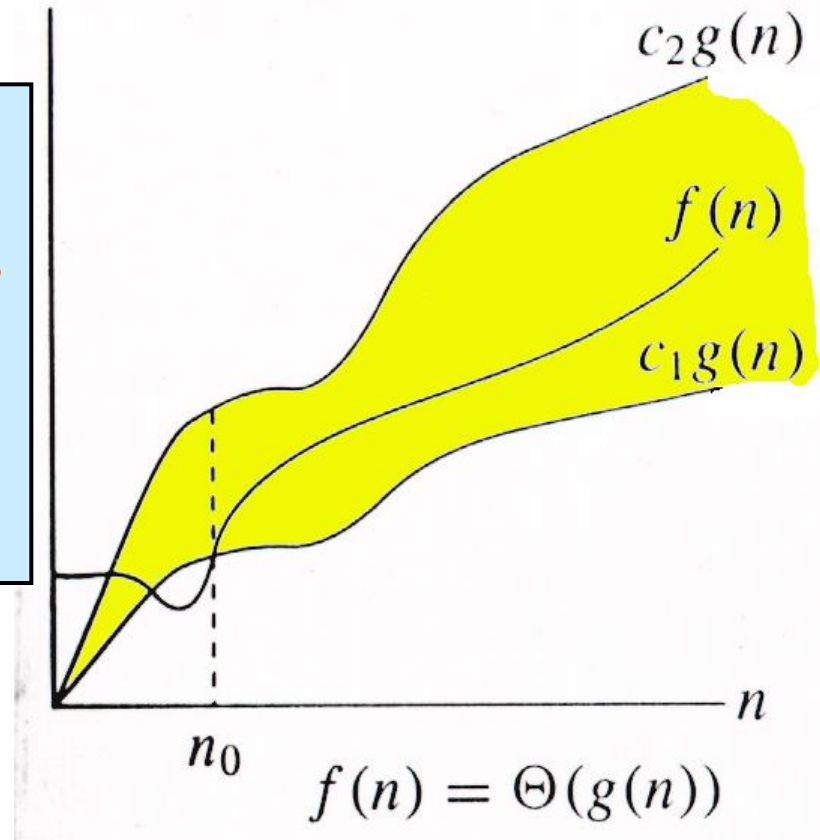
$f(n) = \Theta(g(n))$

$g(n)$ **is an** *asymptotically tight bound* **for** $f(n)$.

Θ-notation

For function $g(n)$, we define $\Theta(g(n))$,
big-Theta of n, as the set:

$\Theta(g(n)) = \{f(n) :$

$\exists$ **positive constants** c_1, c_2, **and** n_0,
such that $\forall n \geq n_0$,

we have $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$

$\}$

$f(n) = \Theta(g(n))$

Technically, $f(n) \in \Theta(g(n))$.
Older usage, $f(n) = \Theta(g(n))$.
I'll accept either…

$f(n)$ and $g(n)$ are nonnegative, for large n.

Example

$\Theta(g(n)) = \{f(n) : \exists$ **positive constants** c_1, c_2, **and** n_0, **such that** $\forall n \geq n_0, \quad 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)\}$

- $10n^2 - 3n = \Theta(n^2)$

- What constants for n_0, c_1, and c_2 will work?

- Make c_1 a little smaller than the leading coefficient, and c_2 a little bigger.

- ***To compare orders of growth, look at the leading term.***

- Exercise: Prove that $n^2/2 - 3n = \Theta(n^2)$

Example

$$\Theta(g(n)) = \{f(n) : \exists \text{ positive constants } c_1, c_2, \text{ and } n_0,$$
$$\text{such that } \forall n \geq n_0, \quad 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)\}$$

- Is $3n^3 \in \Theta(n^4)$??

- How about $2^{2n} \in \Theta(2^n)$??

CSE 270 Design and Analysis of Algorithm, Prof. Jayanthi. G, SRET,

O-notation

For function $g(n)$, we define $O(g(n))$, big-O of n, as the set:
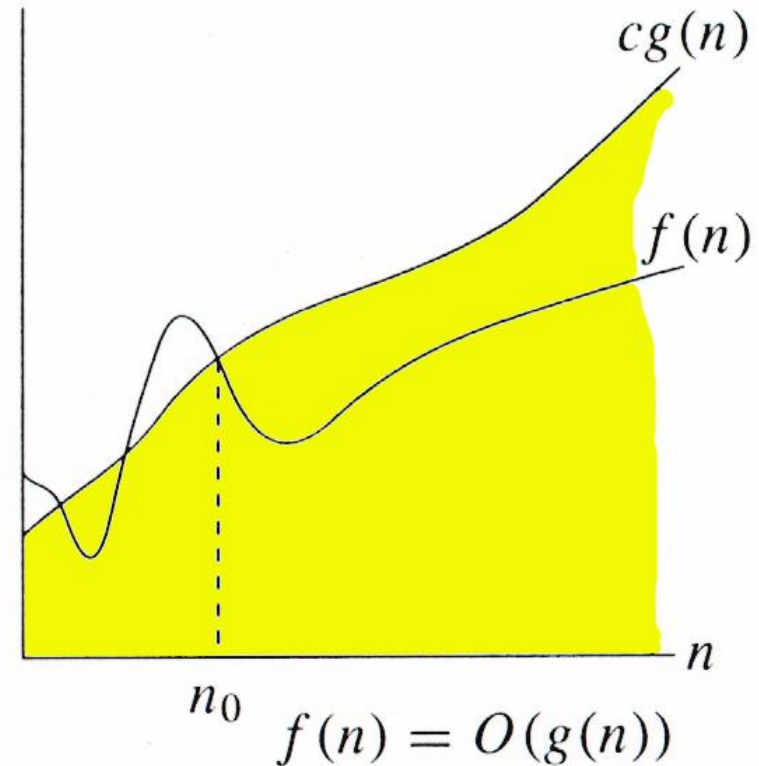
$O(g(n)) = \{f(n) :$
$\exists$ **positive constants** c **and** n_0, **such that** $\forall n \geq n_0$,

we have $0 \leq f(n) \leq cg(n) \}$

Intuitively: Set of all functions whose *rate of growth* is the same as or lower than that of $g(n)$.

$f(n) = O(g(n))$

$g(n)$ **is an** *asymptotic upper bound* **for** $f(n)$.

$f(n) = \Theta(g(n)) \Rightarrow f(n) = O(g(n))$.
$\Theta(g(n)) \subset O(g(n))$.

Examples

$O(g(n)) = \{f(n) : \exists$ **positive constants** c **and** n_0, **such that** $\forall n \geq n_0$, **we have** $0 \leq f(n) \leq cg(n)\}$

- Any linear *function an + b* is in $O(n^2)$. **How?**

- *an + b, a=25 it is value*

- *an^3+b = O(n^3)*

- a = 25, N=5

- Show that $3n^3 = O(n^4)$ for appropriate c and n_0.

CSE 270 Design and Analysis of Algorithm,
Prof. Jayanthi. G, SRET,

Ω -notation

For function $g(n)$, we define $\Omega(g(n))$, big-Omega of n, as the set:

$\Omega(g(n)) = \{f(n) :$
$\exists$ **positive constants c and n_0, such that $\forall n \geq n_0$,**

we have $0 \leq cg(n) \leq f(n)\}$

Intuitively: Set of all functions whose *rate of growth* is the same as or higher than that of $g(n)$.

$f(n)$

$cg(n)$

n_0

$f(n) = \Omega(g(n))$

n

$g(n)$ is an *asymptotic lower bound* for $f(n)$.

$f(n) = \Theta(g(n)) \Rightarrow f(n) = \Omega(g(n)).$
$\Theta(g(n)) \subset \Omega(g(n)).$

CSE 270 Design and Analysis of Algorithm,
Prof. Jayanthi. G, SRET,

Example
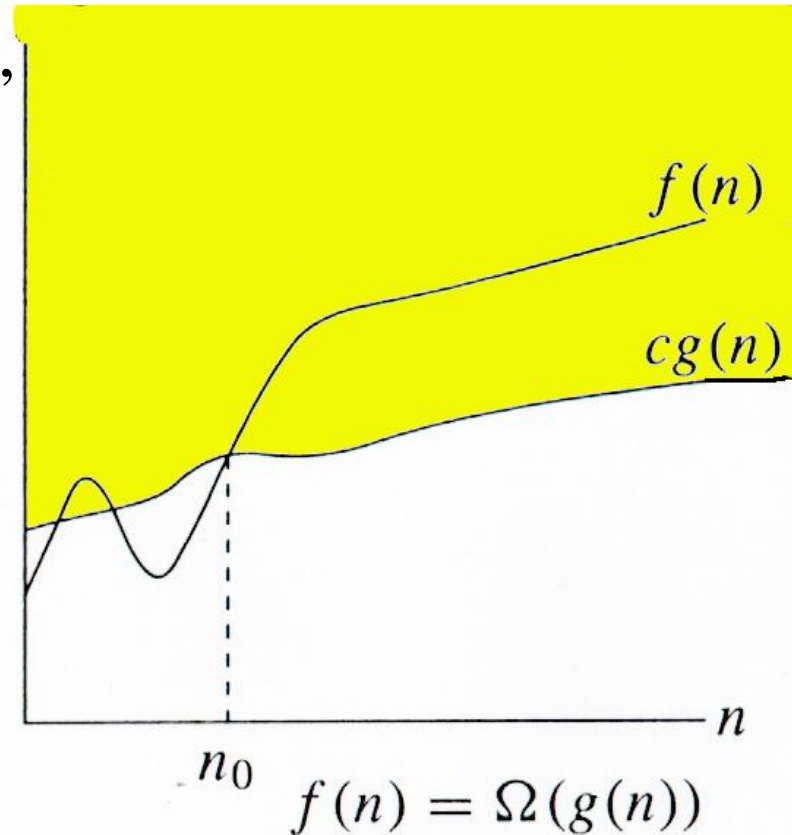
$\Omega(g(n)) = \{f(n) : \exists$ positive constants c and n_0, such that $\forall n \geq n_0$, we have $0 \leq cg(n) \leq f(n)\}$

- $\sqrt{n} = \Omega(\lg n)$. Choose c and n_0.

CSE 270 Design and Analysis of Algorithm,
Prof. Jayanthi. G, SRET,

Relations Between Θ, *O,* Ω

$f(n) = \Theta(g(n))$ $f(n) = O(g(n))$ $f(n) = \Omega(g(n))$

CSE 270 Design and Analysis of Algorithm,
Prof. Jayanthi. G, SRET,
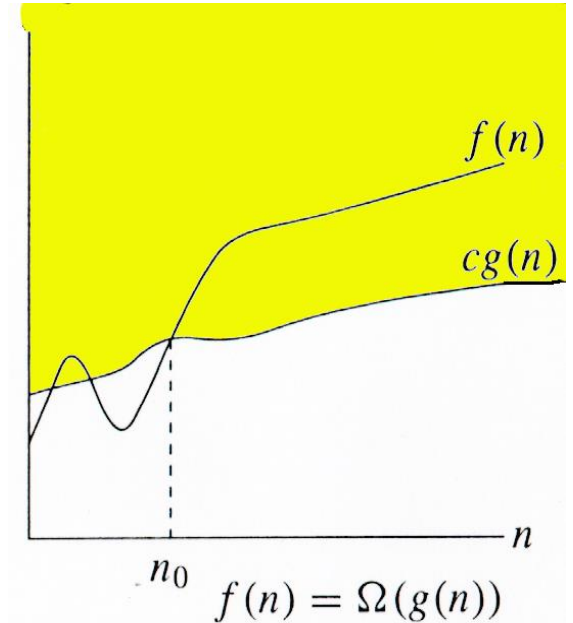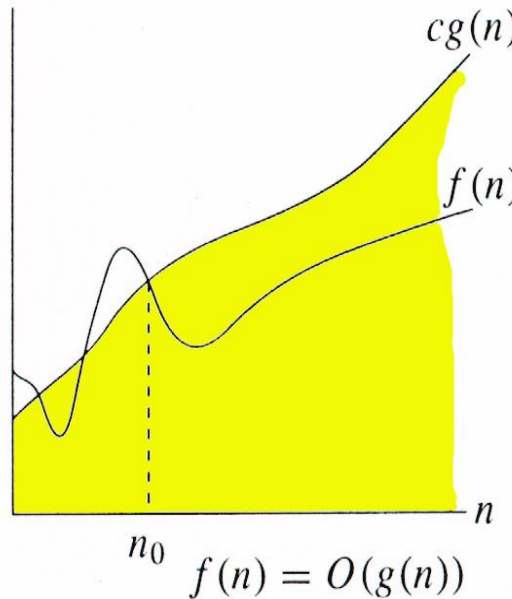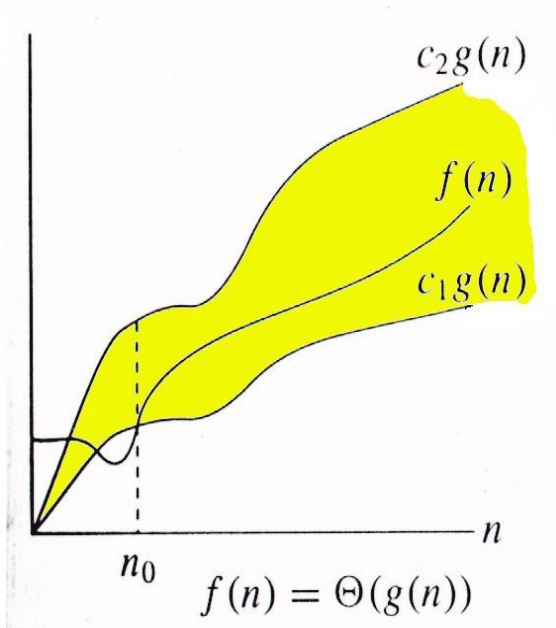
Relations Between Θ, Ω, O

Theorem : For any two functions $g(n)$ and $f(n)$,
$f(n) = \Theta(g(n))$ iff
$f(n) = O(g(n))$ **and** $f(n) = \Omega(g(n))$.

- I.e., $\Theta(g(n)) = O(g(n)) \cap \Omega(g(n))$

- In practice, asymptotically tight bounds are obtained from asymptotic upper and lower bounds.

Comparison of Functions

$$f \leftrightarrow g \ \approx \ a \leftrightarrow b$$

$$f(n) = O(g(n)) \ \approx \ a \ \leq \ b$$

$$f(n) = \Omega(g(n)) \ \approx \ a \ \geq \ b$$

$$f(n) = \Theta(g(n)) \ \approx \ a \ = \ b$$

$$f(n) = o(g(n)) \ \approx \ a \ < \ b$$

$$f(n) = \omega(g(n)) \ \approx \ a \ > \ b$$

CSE 270 Design and Analysis of Algorithm,
Prof. Jayanthi. G, SRET,

Limits

- $\lim\limits_{n \to \infty} [f(n) \, / \, g(n)] = 0 \Rightarrow f(n) \in o(g(n))$

- $\lim\limits_{n \to \infty} [f(n) \, / \, g(n)] < \infty \Rightarrow f(n) \in O(g(n))$

- $0 < \lim\limits_{n \to \infty} [f(n) \, / \, g(n)] < \infty \Rightarrow f(n) \in \Theta(g(n))$

- $0 < \lim\limits_{n \to \infty} [f(n) \, / \, g(n)] \Rightarrow f(n) \in \Omega(g(n))$

- $\lim\limits_{n \to \infty} [f(n) \, / \, g(n)] = \infty \Rightarrow f(n) \in \omega(g(n))$

- $\lim\limits_{n \to \infty} [f(n) \, / \, g(n)]$ undefined $\Rightarrow$ can't say

Properties

- **Transitivity**

$f(n) = \Theta(g(n))$ & $g(n) = \Theta(h(n)) \Rightarrow f(n) = \Theta(h(n))$
$f(n) = O(g(n))$ & $g(n) = O(h(n)) \Rightarrow f(n) = O(h(n))$
$f(n) = \Omega(g(n))$ & $g(n) = \Omega(h(n)) \Rightarrow f(n) = \Omega(h(n))$
$f(n) = o(g(n))$ & $g(n) = o(h(n)) \Rightarrow f(n) = o(h(n))$
$f(n) = \omega(g(n))$ & $g(n) = \omega(h(n)) \Rightarrow f(n) = \omega(h(n))$

- **Reflexivity**

$f(n) = \Theta(f(n))$, x belongs to a set A, then x is reflexive
$f(n) = O(f(n))$
$f(n) = \Omega(f(n))$

19 October 2020

CSE 270 Design and Analysis of Algorithm,
Prof. Jayanthi. G, SRET,

15

Properties

- **Symmetry**

$$f(n) = \Theta(g(n)) \text{ iff } g(n) = \Theta(f(n))$$

- **Complementarity**

$$f(n) = O(g(n)) \text{ iff } g(n) = \Omega(f(n))$$

$$f(n) = o(g(n)) \text{ iff } g(n) = \omega((f(n))$$

Common Functions

CSE 270 Design and Analysis of Algorithm,
Prof. Jayanthi. G, SRET,

Monotonicity

- $f(n)$ is
  - **monotonically increasing** if $m \leq n \Rightarrow f(m) \leq f(n)$.
  - **monotonically decreasing** if $m \geq n \Rightarrow f(m) \geq f(n)$.
  - **strictly increasing** if $m < n \Rightarrow f(m) < f(n)$.
  - **strictly decreasing** if $m > n \Rightarrow f(m) > f(n)$.

CSE 270 Design and Analysis of Algorithm,
Prof. Jayanthi. G, SRET,

Exponentials

- **Useful Identities:**

$$a^{-1} = \frac{1}{a}$$

$$(a^m)^n = a^{mn}$$

$$a^m a^n = a^{m+n}$$

- **Exponentials and polynomials**

$$\lim_{n \to \infty} \frac{n^b}{a^n} = 0$$

$$\Rightarrow n^b = o(a^n)$$

Logarithms

$x = \log_b a$ is the
  exponent for $a = b^x$.

Natural log: $\ln a = \log_e a$

Binary log: $\lg a = \log_2 a$

$\lg^2 a = (\lg a)^2$

$\lg \lg a = \lg (\lg a)$

$$a = b^{\log_b a}$$

$$\log_c (ab) = \log_c a + \log_c b$$

$$\log_b a^n = n \log_b a$$

$$\log_b a = \frac{\log_c a}{\log_c b}$$

$$\log_b (1/a) = -\log_b a$$

$$\log_b a = \frac{1}{\log_a b}$$

$$a^{\log_b c} = c^{\log_b a}$$

CSE 270 Design and Analysis of Algorithm,
Prof. Jayanthi. G, SRET,

Logarithms and exponentials – Bases

- If the base of a logarithm is changed from one constant to another, the value is altered by a constant factor.

  - **Ex:** $\log_{10} n$ * **$\log_2 10$** $= \log_2 n$.
  - Base of logarithm is not an issue in asymptotic notation.

- Exponentials with different bases differ by a exponential factor (not a constant factor).

  - **Ex:** $2^n = $ **$(2/3)^n$** $*3^n$.

Polylogarithms

- For $a \geq 0,\ b > 0$, $\lim_{n \to \infty} (\lg^a n\ /\ n^b) = 0$, so $\lg^a n = o(n^b)$, and $n^b = \omega(\lg^a n)$
  - Prove using L'Hopital's rule repeatedly

- $\lg(n!) = \Theta(n \lg n)$
  - **Prove using Stirling's approximation (in the text) for $\lg(n!)$.**

Exercise

Express functions in A in asymptotic notation using functions in B.

A   B

$5n^2 + 100n$ $3n^2 + 2$ $A \in \Theta(B)$

$A \in \Theta(n^2), n^2 \in \Theta(B) \Rightarrow A \in \Theta(B)$

$\log_3(n^2)$ $\log_2(n^3)$ $A \in \Theta(B)$

$\log_b a = \log_c a / \log_c b; A = 2\lg n / \lg 3, B = 3\lg n, A/B = 2/(3\lg 3)$

$n^{\lg 4}$ $3^{\lg n}$ $A \in \omega(B)$

$a^{\log b} = b^{\log a}; B = 3^{\lg n} = n^{\lg 3}; A/B = n^{\lg(4/3)} \rightarrow \infty$ as $n \rightarrow \infty$

$\lg^2 n$ $n^{1/2}$ $A \in o(B)$

$\lim_{n \rightarrow \infty} (\lg^a n / n^b) = 0$ (here $a = 2$ and $b = 1/2) \Rightarrow A \in o(B)$

Summations – Review

CSE 270 Design and Analysis of Algorithm,
Prof. Jayanthi. G, SRET,

Review on Summations

- Why do we need summation formulas?

  For computing the running times of iterative constructs (loops). (CLRS – Appendix A)

  Example: Maximum Subvector

  Given an array $A[1\ldots n]$ of numeric values (can be positive, zero, and negative) determine the subvector $A[i\ldots j]$ ($1 \leq i \leq j \leq n$) whose sum of elements is maximum over all subvectors.

1	-2	2	2

$1-2+2+2 = 3$ is the value

$1+2+2=5$ is the maximum over all subvectors

$1,2,2$

19 October 2020

CSE 270 Design and Analysis of Algorithm, Prof. Jayanthi. G, SRET,

25

Review on Summations

MaxSubvector*(A, n)*
   maxsum ← 0;
   for *i* ← 1 **to** *n*
    do for *j = i* to *n*
       sum ← 0
      for *k* ← *i* to *j*
        do *sum* += *A[k]*
        sum = sum +A[k]
      maxsum ← max(*sum, maxsum*)
   return maxsum

◆ $T(n) = \sum_{i=1}^{n} \sum_{j=i}^{n} \sum_{k=i}^{j} 1$

◆ NOTE: This is not a simplified solution. What *is* the final answer?

Review on Summations

- **Constant Series:** For integers a and b, $a \leq b$,

$$\sum_{i=a}^{b} 1 = b - a + 1$$

- **Linear Series (Arithmetic Series):** For $n \geq 0$,

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

- **Quadratic Series:** For $n \geq 0$,

$$\sum_{i=1}^{n} i^2 = 1^2 + 2^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}$$

Review on Summations

- **Cubic Series:** For $n \geq 0$,

$$\sum_{i=1}^{n} i^3 = 1^3 + 2^3 + \cdots + n^3 = \frac{n^2(n+1)^2}{4}$$

- **Geometric Series:** For real $x \neq 1$,

$$\sum_{k=0}^{n} x^k = 1 + x + x^2 + \cdots + x^n = \frac{x^{n+1} - 1}{x - 1}$$

For $|x| < 1$,

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1-x}$$

Review on Summations

- **Linear-Geometric Series:** For $n \geq 0$, real $c \neq 1$,

$$\sum_{i=1}^{n} ic^i = c + 2c^2 + \cdots + nc^n = \frac{-(n+1)c^{n+1} + nc^{n+2} + c}{(c-1)^2}$$

- **Harmonic Series:** nth harmonic number, $n \in I^+$,

$$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$$

$$= \sum_{k=1}^{n} \frac{1}{k} = \ln(n) + O(1)$$

CSE 270 Design and Analysis of Algorithm,
Prof. Jayanthi. G, SRET,

Review on Summations

- **Telescoping Series:**

$$\sum_{k=1}^{n} a_k - a_{k-1} = a_n - a_0$$

- **Differentiating Series:** For $|x| < 1$,

$$\sum_{k=0}^{\infty} kx^k = \frac{x}{(1-x)^2}$$

Review on Summations

- **Approximation by integrals:**
  - For monotonically increasing $f(n)$

$$\int\limits_{m-1}^{n} f(x)dx \leq \sum_{k=m}^{n} f(k) \leq \int\limits_{m}^{n+1} f(x)dx$$

  - For monotonically decreasing $f(n)$

$$\int\limits_{m}^{n+1} f(x)dx \leq \sum_{k=m}^{n} f(k) \leq \int\limits_{m-1}^{n} f(x)dx$$

- **How?**

CSE 270 Design and Analysis of Algorithm,
Prof. Jayanthi. G, SRET,

Review on Summations

- ***n*th harmonic number**

$$\sum_{k=1}^{n} \frac{1}{k} \geq \int_{1}^{n+1} \frac{dx}{x} = \ln(n+1)$$

$$\sum_{k=2}^{n} \frac{1}{k} \leq \int_{1}^{n} \frac{dx}{x} = \ln n$$

$$\Rightarrow \sum_{k=1}^{n} \frac{1}{k} \leq \ln n + 1$$