CSC006P1M: Design and Analysis of Algorithms
Lecture 05 *(Analysis of Algorithms)*

Sumit Kumar Pandey

August 24, 2022

RAM Model

RAM Model

- RAM model means **Random-Access Machine** model.
- Algorithms can be measured in a **machine-independent** way using the RAM model.
- This model assumes a **single** processor.
- In this model, instructions are executed one after the another, with **no concurrent** operations.

RAM Model

Assumptions of RAM Model

1. Each simple operations takes constant amount of time. Simple operations consist of
   1. Arithmetic $\rightarrow$ add, subtract, multiply, divide, remainder, floor, ceiling etc.
   2. Data movement $\rightarrow$ load, store, copy.
   3. Control $\rightarrow$ conditional and unconditional branch, subroutine call and return.
2. Loops and subroutines are not simple operations.
3. There is no shortage of memory. The RAM model takes no notice of whether an item is in cache or on the disk, which simplifies the analysis.

Input Size

- Input size depends on the problem being studied.
- For many problems, such as sorting, the most natural measure is the **number of inputs** in the input.
- For many problems, such as multiplying two integers, the best measure of input size is the **total number of bits** needed to represent the input in ordinary binary notation.
- Sometimes, it is appropriate to describe the size of the input with **two numbers rather than one**. For instance, if the input to an algorithm is a graph, the input size can be described by the number of **vertices** and **edges** in a graph.
- It is generally described which input size measure is being used with the problem we study.

Running Time

- The running time of an algorithm on a **particular input** is the number of primitive operations or “steps” executed.
- It is convenient to define the notion of step so that it is as machine-independent as possible.

Example

Example

```
Input: An array A of integers
Output: A
begin
    for j := 2 to A.length do
        key := A[j];
        i := j − 1;
        while i > 0 and A[i] > key do
            A[i + 1] := A[i];
            i := i − 1;
        A[i + 1] := key;
end
```

Example

Input: An array A of integers
Output: A

	cost	times
begin		
for $j := 2$ to A.length do	c_1	n
$key := A[j]$;	c_2	$n-1$
$i := j-1$;	c_3	$n-1$
while $i > 0$ and $A[i] > key$ do	c_4	$\sum_{j=2}^{n} t_j$
$A[i+1] := A[i]$;	c_5	$\sum_{j=2}^{n}(t_j-1)$
$i := i-1$;	c_6	$\sum_{j=2}^{n}(t_j-1)$
$A[i+1] := key$;	c_7	$n-1$
end		

The total running time $T(n) = c_1 n + c_2(n-1) + c_3(n-1) +$
$$c_4 \sum_{j=2}^{n} t_j + c_5 \sum_{j=2}^{n}(t_j-1) + c_6 \sum_{j=2}^{n}(t_j-1) + c_7(n-1)$$

Example

The running time $T(n)$ has a varying parameter t_j. Observe that $1 \leq t_j \leq j$.

- Best Case: $t_j = 1$.

$$\begin{aligned} T_b(n) &= c_1 n + c_2(n-1) + c_3(n-1) + c_4(n-1) + c_7(n-1) \\ &= (c_1 + c_2 + c_3 + c_4 + c_7)n - (c_2 + c_3 + c_4 + c_7) \end{aligned}$$

  So, $T_b(n) = \Omega(1), \Omega(n), \Theta(n), O(n), O(n^2)$

- Worst Case: $t_j = j$
  $T_w(n) = a_1 n^2 + a_2 n + a_3$ (Find a_1, a_2 and a_3)
  So, $T_w(n) = \Omega(1), \Omega(n), \Omega(n^2), \Theta(n^2), O(n^2), O(n^3)$
  But, $T_w(n) \neq \Theta(n), O(n)$.

Example

Let $T(n)$ be running time of the algorithm discussed in the example. Then,

- $T(n) \stackrel{?}{=} \Omega(1)$
- $T(n) \stackrel{?}{=} \Omega(n)$
- $T(n) \stackrel{?}{=} \Omega(n^2)$
- $T(n) \stackrel{?}{=} \Theta(n)$
- $T(n) \stackrel{?}{=} \Theta(n^2)$
- $T(n) \stackrel{?}{=} O(n)$
- $T(n) \stackrel{?}{=} O(n^2)$
- $T(n) \stackrel{?}{=} O(n^3)$

Example

Let $T(n)$ be running time of the algorithm discussed in the example. Then,

- $T(n) = \Omega(1)$
- $T(n) = \Omega(n)$
- $T(n) \neq \Omega(n^2)$
- $T(n) \neq \Theta(n)$
- $T(n) \neq \Theta(n^2)$
- $T(n) \neq O(n)$
- $T(n) = O(n^2)$
- $T(n) = O(n^3)$

Some More Functions

Polynomials

Given a non-negative integer d, a polynomial in n of degree d is a function of $p(n)$ of the form

$$p(n) = \sum_{i=0}^{d} a_i n^i,$$

where $a_0, a_1, \cdots, a_d$ are the coefficients of the polynomial and $a_d \neq 0$.

- A polynomial is asymptotically positive if and only if $a_d > 0$.
- For an asymptotically positive polynomial $p(n)$ of degree d, we have $p(n) = \Theta(n^d)$.
- We say that a function $g(n)$ is **polynomially** bounded if $g(n) = O(n^k)$ for some constant k.

A function $f(n)$ is called exponential function if it is of the form

$$f(n) = ca^n$$

where a, c are positive real numbers and a not equal to 1.

- For all real constants a and b such that $a > 1$,

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

- Therefore, $n^b = o(a^n)$ when $a > 1$.

Logarithms

Logarithm is the inverse operation to exponentiation. For $k > 0$ and $n > 0$, $\log_k n$ is that real number L such that

$$k^L = n$$

We denote "$\log_e$" as "ln" and "$\log_2$" as "lg"

- For any constant $b > 0$,

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

- Therefore, $\lg^a n = o(n^b)$.
- We say that a function $g(n)$ is **polylogarithmically** bounded if $g(n) = O(\lg^k n)$ for some constant k.

Factorials

The notation $n!$ (read "n factorial") is defined for integers $n \geq 0$ as

$$n! = \begin{cases} 1 & \text{if } n = 0, \\ n \cdot (n-1)! & \text{if } n > 0 \end{cases}$$

- $n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right)$[1] (Stirling's approximation)
- $n! = o(n^n)$
- $n! = \omega(2^n)$
- $\lg(n!) = \Theta(n \lg n)$ (use Stirling's approximation)

[1] $f(n) = g(n) + \Theta(n^c)$ means there exists a function $h(n) = \Theta(n^c)$ such that $f(n) = g(n) + h(n)$

Summary

Exponentials $a^n; a > 1$

Polynomials $\vdots$ n^3 $\vdots$ n^2 $\vdots$ n

Polylogarithmics $\lg^a n; a > 0$

Constants

Thank You