Introduction to Algorithms

Lecture 2

July 24, 2019

Primitive Operations

Operations that can be performed in constant amount of time are known as primitive operations. For ex., addition, comparison, multiplication, division, assignment between 2 small numbers(i.e. numbers that can be stored in 1 computer word) can be performed in constant amount of time.

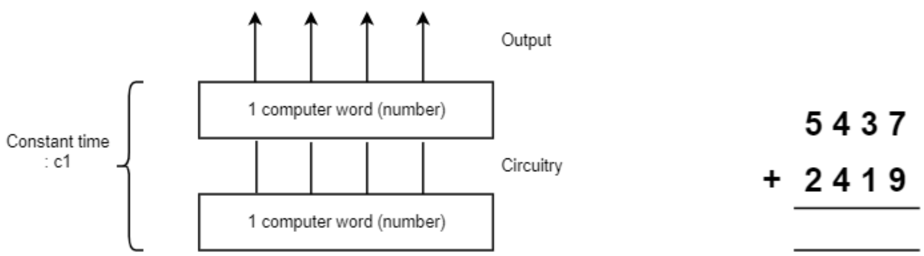

Figure 1: Addition operation between 2 small numbers

Are Loop Operations Primitive Operations?
Loop Operations are not considered as primitive operations as 1 loop operation contains multiple primitive operations and hence cannot be performed in constant amount of time.

```
for i<-1 to n
    x[i]<-y[i]
```

Operations in 1 loop iteration:

S.No.	Operation	Description
1	i=1	Assign i value 1
2	y[i]	Address calculation of y[i] and fetching value
3	x[i]	Address calculation of x[i] and fetching value
4	x[i]=y[i]	Assignment

If numbers are not small?
If operations are performed between numbers that are not small then the operation cannot be completed in constant amount of time.

Representation of a large number
Let X be a large number.

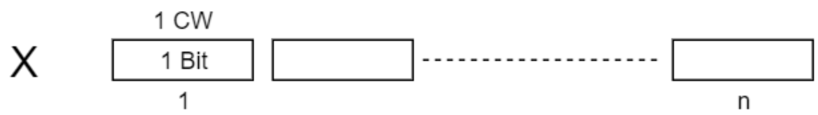

Figure 2: Input size of X is n computer words

- Number of bits taken by X is $b = \log_2 X$
- Since 1 computer word contains 1 bit, number of computer words taken by X is $n = \log_2 X$
- If 1 computer word = 2 bits, then number of bits $b = \log_2 X$ and number of computer words $n = \frac{\log_2 X}{2}$ which is same as $\log_4 X$
- We can write $\frac{\log_2 X}{2}$ as $\log_4 X$:

$$\log_4 X = \frac{\log_b X}{log_b 4} = \frac{\log_b X}{log_b 2^2} = \frac{\log_b X}{2 log_b 2} = \frac{\log_2 X}{2}$$

- Now, if 1 computer word= 3 bits, $n = \log_8 X$

Calculate number of computer words:

Bits in 1 CW	No. of computer words(n)
1	$\log_2 X$
2	$\log_4 X$
3	$\log_8 X$
4	$\log_{16} X$

Result:

So, time taken to perform operation between two large numbers X and Y taking more than 1 computer word to store is
= O[max[$\log_2 X$, $\log_2 Y$]]
= O(n)
= Linear Time
$\neq$ Constant Time

Asymptotic Notations

Notations to analyze and represent an algorithm's run-time performance.

Running Time= No. of primitive operations X Constant Time

1. **Big O Notation**
   Def: *Given functions f(n) and g(n),*
   $$f(n) = O(g(n))$$
   if there exists constants $c > 0$ and $m > 0$ such that for all $n \geq m$, we have,
   $$f(n) \leq c \cdot g(n).$$

   We measure performance for value of large inputs(as n grows higher).
   As soon as n crosses m, c.g(n) leads f(n).

   Graphical Representation:

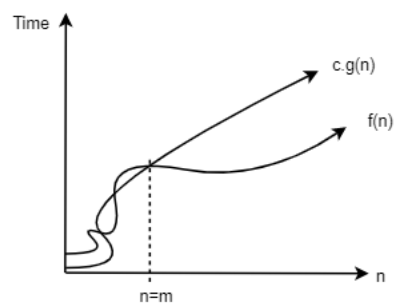

Figure 3: Big O

2. **Theta (θ) Notation**
   Def: *Given functions f(n) and g(n),*
   $$f(n) = \theta(g(n))$$
   if there exists constants $c_1 > 0$, $c_2 > 0$ and $m > 0$ such that for all $n \geq m$, we have,
   $$c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n).$$

   We measure performance for value of large inputs (as n grows higher).
   As soon as n crosses m, the value of f(n) lies above c_1.g(n) and at or below c_2.g(n). In other words, for all n$\geq$ m, f(n) is equal to g(n) to within a constant factor.
   We can also use notation f(n) = g(n).

Graphical Representation:

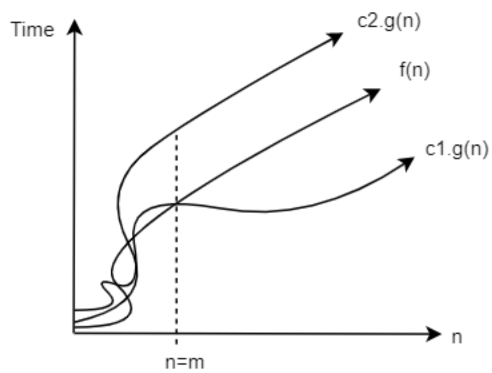

Figure 4: Theta (θ)

3. **Omega (Ω) Notation**
   Def: *Given functions f(n) and g(n),*
   $$f(n) = \Omega(g(n))$$
   if there exists constants $c > 0$ and $m > 0$ such that for all $n \geq m$, we have,
   $$0 \leq c \cdot g(n) \leq f(n).$$

   We measure performance for value of large inputs (as n grows higher).
   As soon as n crosses m, the value of f(n) on or above c.g(n).

   Graphical Representation:

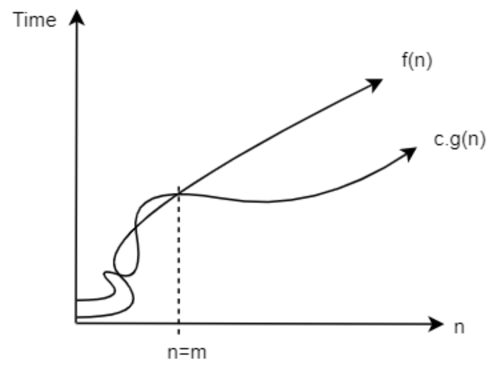

Figure 5: Omega (Ω)

Limit form Definitions for Asymptotic Notations

1. **Big O Notation**

   Def: *Given functions f(n) and g(n), If*

   $$\lim_{x\to\infty}\frac{f(x)}{g(n)} = c \qquad \{0 \le c < \infty\}$$

   $$\textit{then } f(n) \le g(n) \textit{ i.e. } f(n) = O(g(n))\ .$$

   O represents a combination of θ and o

2. **Theta θ Notation**

   Def: *Given functions f(n) and g(n), If*

   $$\lim_{x\to\infty}\frac{f(x)}{g(n)} = c \qquad \{0 < c < \infty\}$$

   $$\textit{then } f(n) = g(n) \textit{ i.e. } f(n) = \theta(g(n))\ .$$

3. **Omega Ω Notation**

   Def: *Given functions f(n) and g(n), If*

   $$\lim_{x\to\infty}\frac{f(x)}{g(n)} = c \qquad \{0 < c \le \infty\}$$

$$\textit{then } f(n) \geq g(n) \textit{ i.e. } f(n) = \Omega(g(n)) \text{ .}$$

Ω represents a combination of θ and ω

4. **Small o Notation**

   Def: *Given functions f(n) and g(n), If*

$$\lim_{x \to \infty} \frac{f(x)}{g(n)} = 0$$

$$\textit{then } f(n) < g(n) \textit{ i.e. } f(n) = o(g(n)) \text{ .}$$

   f(n) is strictly less than g(n)

5. **Small Omega ω Notation**

   Def: *Given functions f(n) and g(n), If*

$$\lim_{x \to \infty} \frac{f(x)}{g(n)} = \infty$$

$$\textit{then } f(n) > g(n) \textit{ i.e. } f(n) = \omega(g(n)) \text{ .}$$

   f(n) is strictly greater than g(n)