Unit 1

* **Fundamentals of Algorithmic Problem Solving**

Space Complexity: Amount of memory it needs to run to completion.

Algorithm: A finite set of instructions to accomplish a particular task.

Must satisfy the following criteria:

(i) have an input

(ii) have an output

(iii) definiteness - each instruction is clear & unambiguous

(iv) finiteness - algorithm terminates after a finite no. of steps

(v) effectiveness - instructions should be basic and effective

* **Types of Algorithms**

(i) sorting — rearrange items of a given list in ascending order

stable algo — preserves relative order of any 2 equal elements in its input

(ii) searching - finding a given value called a search key

(iii) string processing

(iv) graph problems - many are np hard, like the traveling salesman problem & graph coloring problem

(v) combinatorial problems - find a combinatorial object - such as a permutation, combination or subset that satisfies certain constraints and has some desired property

(vi) Geometric problems - deal w/ points, lines & polygons. eg, closest-pair problem & the convex hull problem

(vii) numerical problems - solving equations & systems of equations, computing definite integrals, evaluating fns etc

* Measuring Input Size

(i) sorting, searching, finding smallest element in list = n

(ii) polynomial evaluation : degree / no. of coefficients

(iii) matrix mult : $\max(m,n)$

(iv) algorithms involving properties of nos : $b = \lfloor \log_2 n \rfloor + 1$
(checking if prime) no. of bits

* Worst Case Efficiency : where for an input of size n, the algorithm takes the longest to run

* Best - Case Efficiency : where for an input of size n, the algorithm runs the fastest.

* Asymptotic Notation

O- notation - A fn is said to be $O(g(n))$ denoted by $t(n) \in O(g(n))$ if $t(n)$ is bounded above by some constant multiple of $g(n)$

$$t(n) \le c g(n)$$

Ω -notation - $t(n)$ is bounded below by some +ve constant multiple of $g(n)$

$$t(n) \ge c g(n)$$

Θ - notation : If $t(n)$ is bounded both above & below by some +ve constant multiples of $g(n)$

$$c_2 g(n) \le t(n) \le c_1 g(n)$$

* Property : $t_1(n) \in O(g_1(n))$ &
$t_2(n) \in O(g_2(n))$

then $t_1(n) + t_2(n) \in O(\max\{g_1(n), g_2(n)\})$

* Limits for comparing orders of growth

Find

$$\lim_{n\to\infty} \frac{t(n)}{g(n)}$$

L' Hopital's Rule: $\lim_{n\to\infty} \frac{t(n)}{g(n)} = \lim_{n\to\infty} \frac{t'(n)}{g'(n)}$

Stirling's Formula: $n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$

Ex1 Compare order of growth of $\frac{1}{2}n(n-1)$ & n^2

$$\lim_{n\to\infty} \frac{\frac{1}{2}(n^2 - n)}{n^2}$$

$$\frac{1}{2}\lim_{n\to\infty} 1 - \frac{1}{n} = \frac{1}{2}$$

Ex2 Compare orders of growth of $\log_2 n$ and $\sqrt{n}$

$$\lim_{n\to\infty} \frac{\log n}{\sqrt{n}} = \frac{\frac{1}{n \cdot 1}}{\frac{1}{2\sqrt{n}}} = \frac{2\sqrt{n}}{n} = 0 \quad ?$$

Ex3 Compare growth of $n!$ and 2^n

$$\lim_{n\to\infty} \frac{n!}{2^n} = \lim_{n\to\infty} \frac{\sqrt{2\pi n}\left(\frac{n}{e}\right)^n}{2^n} = \frac{\sqrt{2\pi n} \cdot n^n}{2^n \cdot e^n}$$

$$= \infty$$

Exercises : From the following inequalities, check if they are correct.

$t(n) \leq c\,g(n)$

a. ~~$6n^2 - 8n = \Theta(n^2)$~~

$100n + 5 \in O(n^2)$

$100n \leq n^2 \Rightarrow$ True

Ex2 $n^2 + 10n \in O(n^2)$

→ max power $= n^2$ $\Rightarrow \in O(n^2)$

Ex3 $5n^2 \in O(n^2)$

$c = 5$ $n^2 \in O(n^2)$ $\Rightarrow \in O(n^2)$

Ex4 $\frac{n(n-1)}{2} \in O(n^2)$

$= \frac{1}{2}(n^2 - n)$

$\frac{n^2}{2} - \frac{n}{2}$ $\frac{n^2}{2} \leq n^2$ $c = \frac{1}{2}$

Ex5 Order of $3n+2$ $= O(n)$

Order of $10n^2 + 4n + 2$ $=$ $O(n^2)$

Ex6 T or F : $\frac{n(n+1)}{2} \in O(n^3)$ $= T$

b. $\frac{n(n+1)}{2} \in O(n^2)$ $= T$

c. $\frac{n(n+1)}{2} \in \Theta(n^3)$ $= F$

d. $\frac{n(n+1)}{2} \in \Omega(n)$ $= ~~F~~$ T

* Mathematical Analysis for non-recursive algorithms

Ex1

```
fun(m,n){
    n <- n+1
    m <- m+1
}
```

$= \Theta(n)$

Ex2

```
fun(n){
    for (i <- 0 to n-1){
        print(i)
    }
}
```

$= \Theta(n)$

Ex3

```
fun (k ){
    max = k[0]
    for  i<-1  to  n-1 {
            if  k[i] > max
                max <-k [i]
    }   }
```

$\Theta(n)$

Ex4

```
search (k, key){
    found <- False
    i <-0
    n <- length(k)
    while ( found== False AND  i<=n-1) {
        if (k[i]==key)
            found =true
            i <- i+1
    }   }
```

$O(n)$

Ex5

```
fun(n) {
    i=1
    while (i<=n-1){
            i <- i*2 }
}   }
```

$O(n \log n)$

Ex6

```
fun(n) {
    for (i <-0 to n-1) {
        for(j <-0 to n-1 ) {
            print(i,j)
}   }   }
```

$\Theta(n^2)$

Ex7 fun(m,n) {

for (i ← 0 to n-1) {

for (j ← 0 to m-1) {

print (i, j)

$\Theta(mn)$

Ex8 fun(A, B) {

n = len(A) = len(B)

for (i ← 0 to n-1) {

for (j ← 0 to n-1) {

if (A[i] == B[j])

return false

} } }

$\Omega(1)$

$O(n^2)$

Ex9 fun(A) {

n = len(A)

for (i ← 0 to n-1) {

for (j ← 0 to n-1) {

for (k ← 0 to n-1) {

print (i, j, k)

$\Theta(n^3)$

Ex10 MaxElement

maxval ← A[0]

for i ← 1 to n-1 do

if A[i] > maxval

maxval ← A[i]

return maxval

Soln $c(n) = \sum_{i=1}^{i=n-1} (1) = n-1-1+1 \in \Theta(n)$

Ex11 for $i \leftarrow 0$ to $n-2$ do

  for $j \leftarrow i+1$ to $n-1$ do

    if $A[i] = A[j]$

      return false

return true

Soln:

$$\sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1$$

$$= \sum_{i=0}^{n-2} \left[(n-1) - (i+1) + 1\right]$$

$$= \sum_{i=0}^{n-2} \left[n-1-i-\cancel{1}+\cancel{1}\right]$$

$$= \sum_{i=0}^{n-2} \left[n-1-i\right]$$

$$= \sum_{i=0}^{n-2} (n-1) - \sum_{i=0}^{n-2} (i) = (n-1)\sum_{i=0}^{n-2} (1) - \frac{(n-2)(n-1)}{2}$$

$$= (n-1)\{n-2+1\} - \frac{(n-2)(n-1)}{2}$$

$$= (n-1)^2 - \frac{(n-1)(n-2)}{2}$$

$$= (n-1)\left\{1 - \frac{n-2}{2}\right\}$$

$$= (n-1)\left\{\frac{2-n+2}{2}\right\} = \frac{n(n-1)}{2}$$

$$= \frac{1}{2}n^2 \in \Theta(n^2)$$

Ex12 Matrix multiplication

for $i \leftarrow 0$ to $n-1$ do

  for $j \leftarrow 0$ to $n-1$ do

    $C[i,j] \leftarrow 0.0$

    for $k \leftarrow 0$ to $n-1$ do

      $C[i,j] \leftarrow C[i,j] + A[i,k] * B[k,j]$

Solution $\sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} (1)$

$= \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} n \cdot 1 \cdot 1$

$= n^3$ //

Ex13 Decimal to Binary?

```
count ← 1
while n > 1 do
    count ← count + 1
    n ← n/2
return count
```

Soln value of n halved for every rep of loop $\Rightarrow$ $\log n$

base case 1, if n = 0

* Recursive Recurrence Relations

① void fn (int n) { $\longrightarrow T(n)$

if (n>0) {

print (n) $\longrightarrow 1$

Test (n-1) $\longrightarrow T(n-1)$

}

}

$T(n-1) = T(n-2) + 1$

$T(n-2) = T(n-3) + 1$

$T(n) = T(n-1)+1$

sub for $T(n-1)$

$T(n) = [T(n-2) + 1] + 1$

$T(n) = T(n-2) + 2.$

$= [T(n-3) + 1] + 2$

$= T(n-3) + 3$

in general

$T(n) = T(n-k) + k$

$n-k = 0$

$n = k$

$T(n) = T(0) + n$

$= 1+n \Rightarrow O(n)$

② void Test (int n) { $\longrightarrow T(n)$

if (n>0) {

for (i=0, i<n ; i++) {

print(n) $\longrightarrow n$

}

test (n-1) $\longrightarrow T(n-1)$

}

}

$T(n) = T(n-1) + n$ $\qquad\qquad$ $T(n-1) = T(n-2) + n-1$

sub $T(n-1)$ $\qquad\qquad$ $T(n-2) = T(n-3) + n-2$

$= T(n-2) + (n-1) + n$

sub $T(n-2)$

$= T(n-3) + (n-2) + (n-1) + n$

In general

$T(n) = T(n-k) + \cancel{T}(n-(k-1)) + \cancel{T}(n-(k-2)) + \cdots + n$

$n-k=0$

$n=k$

$T(n) = T(0) + \cancel{T}(n-n+1) + \cancel{T}(n-n+2) + \cdots n$

$= T(0) + \cancel{T}(1) + \cancel{T}(2) + \cdots$

$= T(0) + 1 + 2 + \cdots n-1 + n$

$= T(0) + \frac{n(n+1)}{2} \qquad = O(n^2)$ //

(3) void Test (int n) { $\longrightarrow T(n)$

if (n>0)

for (i=1; i<n; i=i*2) $\Big\}$ $\log n$

print (i)

Test (n-1) $\longrightarrow T(n-1)$

$T(n) = T(n-1) + \log n$ $\qquad\qquad$ $T(n-1) = T(n-2) + \log(n-1)$

$= T(n-2) + \log(n-1) + \log n$ $\qquad\qquad$ $T(n-2) = T(n-3) + \log(n-2)$

$= T(n-3) + \log(n-1) + \log(n-2) + \log n$

$n-k=0$

$n=k$

$T(0) + \log n!$

$= 1 + \log n!$ $\quad = O(n \log n)$ //

(4) Test (int n) { $\longrightarrow T(n)$

if (n>0) {

print(n) $\longrightarrow 1$

Test(n-1) $\longrightarrow T(n-1)$

Test(n-1) $\longrightarrow T(n-1)$

}

}

$T(n-1) = 2T(n-2)+1$

$T(n) = 2T(n-1) + 1$ $\qquad T(n-2) = 2T(n-3)+1$

$= 2\cdot\{2T(n-2) + 1\} + 1$

$= 2\{2\{2T(n-3)+1\}+1\}+1$

$= 2^k T(n-k) + 2^{k-1} + 2^{k-2} + \dots 2^2 + 2 + 1.$

$n-k=0$

$= 2^n T(0) + 2^{n-1} + 2^{n-1} + \dots 2^2 + 2 + 1$

$= 2^n T(0) + 1 + 2 + 2^2 + \dots 2^{n-1}$

$= 2^n T(0) + \cancel{n(n+1)(2} \; 2^k - 1$

$= 2^n T(0) + 2^k - 1$

$= O(2^n)$ //

(5) Test (int n) { $\longrightarrow T(n)$

if (n > 1) {

print(n) $\longrightarrow 1$

Test(n/2) $\longrightarrow T(n/2)$

}

}

$n=1,\ T(1) = 1$

$T(n) = T(n/2)+1$

$T(n/2) = T(n/4)+1$

$= (T(n/4)+1)+1$

$T(n/4) = T(n/8+1)$

$= T(n/8)+1+1+1$

⋮

$T(n)\ \ T(n/2^k)+k$

$\dfrac{n}{2^k} = 1 \qquad n = 2^k$

$k = \log_2 n$

$T(n) = T(1) + \log n$

$\boxed{O(\log n)}$

(6) $T(n) = T(n/2)+n$

$T(n/2) = T(n/4) + n/2$

$T(n/4) = T(n/8) + n/4$

$= T(n/4) + n + n/2$

$= T(n/8) + n/4 + n/2 + n$

$= T(n/2^k) + n/2^{k-1} + n/2^{k-2} + \dots n/2 + n/1$

$n/2^k = 1 \qquad n = 2^k \qquad k = \log n$

$= T(1) + n\{1 + \frac{1}{2} + \underbrace{\frac{1}{2^2}}_{\approx 1} + \dots \frac{1}{2^{k-1}}\}$

$= T(1) + n \qquad = O(n)$

Unit - 1

Solving Recurrence Relations

① $T(n) = T\left(\frac{n}{2}\right) + c \qquad T(1) = 1$

$T(n/2) = T(n/4) + c$

$T(n/4) = T(n/8) + c$

$\Rightarrow T(n) = T(n/4) + 2c$

$= T(n/8) + 3c$

$= T(n/2^3) + 3c$

$= T(n/2^k) + kc$

$\frac{n}{2^k} = 1$

$n = \cancel{2^{k}}\ 2^k$

$\Rightarrow k = \log n$

$T(n) = T(1) + \log n \cdot c$

$= 1 + c \cdot \log n$

$\boxed{T(n) = \log n}$

(2) $T(n) = n * T(n-1)$

$T(1) = 1$

$T(n-1) = (n-1) * T(n-2)$

$T(n-2) = (n-2) * T(n-3)$

$$T(n) = n * (n-1) * T(n-2)$$

$$= n * (n-1) * (n-2) * T(n-3)$$

$$= \left(n * (n-1) * (n-2) \times \ldots \; T(n-(n-1))\right)$$

$$= \left(n * (n-1) * (n-2) * \ldots T(1)\right)$$

$$= n * (n-1) * (n-2) * \ldots 1$$

$$= n * n\left(1-\frac{1}{n}\right) * n\left(1-\frac{2}{n}\right) * \ldots$$

$$= n^n$$

$O(n^n)$

(3) $T(n) = 2T\left(\frac{n}{2}\right) + n$

$T(n/2) = 2T\left(\frac{n}{2^2}\right) + \frac{n}{2}$

$T(n/4) = 2T\left(\frac{n}{2^3}\right) + \frac{n}{2^2}$

$$T(n) = 2\left(2T\left(\frac{n}{2^2}\right) + \frac{n}{2}\right) + n$$

$= 2^2 T(n/2^2) + n + n$

$= 2^2 T(n/2^2) + 2n$

$= 2^2 \{ 2T(n/2^3) + n/4 \} + 2n$

$= 2^3 \{ T(n/2^3) + 2n + 2n$

$= 2^3 T(n/2^3) + 3n$

$= 2^k T(n/2^k) + kn$

$\Rightarrow \frac{n}{2^k} = 1$

$n = 2^k \qquad k = \log n$

$= n\,T(1) + n\log n$

$O(n\log n)$

(4) $T(n) = T(n-1) + \log n$

$T(n-1) = T(n-2) + \log(n-1)$

$T(n-2) = T(n-3) + \log(n-2)$

$T(n) = (T(n-2) + \log(n-1)) + \log n$

$T(n) = T(n-3) + \log(n-2) + \log(n-1) + \log(n)$

$\vdots$

$T(n) = T(n-k) + \log(n-(k-1)) + \log(n-(k-2)) + \cdots$

$n-k=1$

$k=1$

$T(n) = 1 + \log(1) + \log(2) + \log(3) + \dots \log(n)$

$= 1 + \log(1.2.3.\dots n)$

$= 1 + \log(n!) \quad = 1 + \log(n^n)$

$n! \Rightarrow n^n$

$O(\log(n!)) = 1 + n\log n$

$\underline{\underline{O(n\log n)}}$

(5) $T(n) = 8T\left(\frac{n}{2}\right) + n^2$

$a = 8$

$b = 2 \qquad a > n^d$

$d = 2$

$\Rightarrow n^{\log_b a} = n^{\log_2 8}$

$= n^3$

By substitution

$T(n/2) = 8T(n/4) + (n/2)^2$

$T(n/4) = 8T(n/2^3) + (n/4)^2 + \dots$

$T(n) = 8\{8T(n/4) + (n/2)^2\} + n^2$

$= 64T(n/4) + 2n^2 + n^2$

$= 64T(n/4) + 3n^2$

$= 64\left(8T(n/8) + (n/4)^2\right) + 3n^2$

$= \cancel{8}\, 8^3 T(n/8) + 4n^2 + 3n^2$

$= 8^3 T(n/8) + 7n^2$

$= 8^3 \; T(n/8) + 2^2 n^2 + 2^1 n^2 + 2^0 n^2$

$= 8^k \; T\left(n/_{2^k}\right) + 2^{k-1} n^2 + 2^{k-2} n^2 + \ldots$

$\frac{n}{2^k} = 1$

$\underline{k = \log n}$

$= 8 \; 2^{3k} \; T(n/2^k) + 2^{k-1} n^2 + 2^{k-2} n^2 + \ldots$

$= 2 (2^k)^3 (1) + n^2 \{ 2^0 + 2^1 + \ldots 2^{k-1} \}$

$= (n^3)(1) + n^2 \left\{ \frac{1(2^k - 1)}{1} \right\}$

$= n^3 + n^2 (n-1)$

$= n^3 + n^3 - n^2$

$= 2n^3 - n^2$

$\Rightarrow \underline{\underline{O(n^3)}}$

(6) $T(n) = 2T\left(\frac{n}{2}\right) + cn$

$T(n/2) = 2T\left(\frac{n}{4}\right) + \frac{cn}{2}$

$T(n/4) = 2T(n/8) + c(n/4)$

$T(n) = 2 \left\{ 2T(n/4) + \frac{cn}{2} \right\} + cn$

$= 4T(n/4) + cn + cn$

$= 4 \{ 2T(n/8) + c(n/4) \} + cn + cn$

$5T(n/5) + 4c(n/4) + cn + cn$

$= 2^3 T\left(\frac{n}{2^3}\right) + 3cn$

$= 2^k T\left(\frac{n}{2^k}\right) + kcn$

$2^k = n$

$k = \log n$

$= n(1) + \log n \cdot c \cdot n$

$O(n \log n)$

(7) $T(n) = 2T(n-3)$ $\quad T(1) = 1$

$T(n-1) = 2T(n-4)$

$T(n-2) = 2T(n-5)$

$T(n-3) = 2T(n-6)$

$T(n) = 2.2T(n-6)$ $\quad$ -(1)

$T(n-6) = 2T(n-9)$ $\quad$ -(2)

$T(n) = 2.2.2\,T(n-9)$

$= 2^3\, T(n - 3*3)$

$= 2^k\, T(n - 3*k)$

$n - 3k = 1$

$n - 1 = 3k$ $\quad k = \frac{n-1}{3}$

$T(n) = 2^{\frac{n-1}{3}}\, T(1)$

$= 2^{n/3} \cdot 2^{-1/3}$ $\quad = O(2^{n/3})$ //

⑧ $T(n) = T(n-2) - 2 \qquad T(1) = 0$

$T(n-2) = T(n-4) - 2$

$T(n-4) = T(n-6) - 2$

$T(n) = (T(n-4) - 2) - 2$

$= T(n-4) - 2 - 2$

$= (T(n-6) - 2) - 2 - 2$

$T(n) = T(n-6) - 2 - 2 - 2 \qquad = T(n-2*3) - 2*3$

$= T(n-2*k) - 2*k$

$T(1) = 0$

$n - 2k = 0$

$n = 2k$

$= T(1) - n$

$\underline{O(n)}$

??

⑨ $T(n) = 2T(n-2) - n \qquad T(0) = 0$

$T(n-2) = 2T(n-4) - (n-2)$

$T(n-4) = 2T(n-6) - (n-4)$

$T(n) = 2\{2T(n-4) - (n-2)\}$

$= 4T(n-4) - 2(n-2) \qquad = 4T(n-4) - 2n - 4$

$= 8T(n-6) - 4n - 16 - 2n + 4$

$= 4\{2T(n-6) - (n-4)\} - 2(n-2)$

$= 8T(n-6) - 4(n-4) - 2(n-2)$

$= 2^3 T(n-2*3) - 2^2(n-2^2) - 2^1(n-2^1)$

$= 2^3 T(n - 2 \cdot 3) - 2^2 (n - 2^2) - 2^1 (n - 2^1)$

$= 2^k T(n - 2k) - 2^k (n - 2^k) \ldots \quad 2^1 (n - 2^1)$

$T(0) = 0$

$n - 2k = 0$

$n = 2k$

$k = n/2$

$= 2^{n/2} \, T(0) - 2^{n/2} (n - 2^{n/2}) \ldots \quad 2^1 (n - 2^1)$

$= 0 - 2^{\frac{n}{2} \cdot n}$

(14) $T(n) = T\left(\frac{n}{2}\right) - 2 \qquad T(1) = 2$

$T(n/2) = T(n/4) - 2$

$T(n/4) = T(n/8) - 2$

$T(n) = T(n/4) - 2 - 2$

$= T(n/8) - 2 - 2 - 2$

$= T(n/2^k) - 2k$

$n/2^k = 1 \qquad n = 2^k \qquad n = 2^k$

$k = \log n$

$\Rightarrow T(1) = 2$

$T(n) = T(1) - 2\log n$

$= 2 - 2\log n \qquad O(\log n)$

(11) $T(n) = T\left(\frac{n}{5}\right) + n \qquad T(1) = 1$

$T(n/5) = T(n/25) + n/5$

$T(n/25) = T(n/125) + n/25$

$T(n) = T(n/25) + n + \frac{n}{5}$

$= T\left(\frac{n}{125}\right) + \frac{n}{25} + \frac{n}{5} + \frac{n}{1}$

$= T\left(\frac{n}{5^k}\right) + \left(\frac{n}{1} + \frac{n}{5} + \frac{n}{25} + \cdots \frac{n}{5^k}\right)$

$= T\left(\frac{n}{5^k}\right) + n\left(\frac{1}{1} + \frac{1}{5} + \frac{1}{25} + \cdots\right)$

$= T\left(\frac{n}{5^k}\right) + n\left(\frac{1\left((1/5)^k - 1\right)}{(1/5 - 1)}\right)$

$\frac{n}{5^k} = 1 \qquad n = 5^k$

$k = \log_5 n$

~~$= T(1) + n\left(\frac{1(n-1)}{4}\right)$~~

$T(1) + n\left(\frac{1 - \left(\frac{1}{5}\right)^k}{\frac{4}{5}}\right)$

$\frac{T(1) + 5n\left(1 - \frac{1}{5^k}\right)}{4}$

$O(n)$

$T(1) + \frac{5n\left(1 - \frac{1}{n}\right)}{4} \Rightarrow T(1) + \frac{5\cancel{n}\left(\frac{n-1}{\cancel{n}}\right)}{4}$