Algorithm two types

iterative

A ( )
{
  for i = 1 to n
    max (a, b)
}

Recursive

A(n)
{
  if ( )
    A (n/2)
}

→ powerwise both same

→ analysis wise both different.

```
A ( )
{
    int i ;
    for (i = 1 to n)          → n
        printf("BAV");
}
```

$O(n)$

```
A ( )
{
    int i;
    for (i = 1 to n)          → n
        for (j = 1 to n)      → n
            printf("BHU")
```

$O(n^2)$

```
A( )
{
    i=1, s=1
    while (s<=n)
    {
        i++
        s = s+i
        printf("Ravi");
    }
}
```

s	1	3	6	10	15	21	......	$\frac{K(K+1)}{2}$
i	1	2	3	4	5	6	.....	K

$$\frac{K(K+1)}{2} > n$$

$$K^2 + K > n$$

$$K = O(\sqrt{n})$$

```
A ( )
{
  i = 1
  for (i = 1, i^2 <= n ; i++)          i <= √n
     print ("BHU"); ------> √n
```

$O(\sqrt{n})$

For this best, worse and Average case are same.
So, we can write it
$\Theta(\sqrt{n})$.

```
A ( )
{ int i, j, k, n;
  for (i = 1 ; i <= n ; i++)
  {
    for (j = 1 ; j <= i ; i++)
    {
      for (k = 1 ; k <= 100 ; k++)
      {
        print ("BHU");
      }
    }
  }
```

$i=1$	$i=2$	$i=n$
$j=1$ times	$j=2$ time	$j=n$ time
$K=100$ times	$K=2*100$ time	$K=n*100$ times

$$100(1+2+\dots+n)$$
$$=100\left[\frac{n(n+1)}{2}\right]$$
$$=O(n^2)$$

```
A()
{
  int i, j, k, n;
  for( i=1 ; i<=n; i++)
  {
    for( j=1 ; j<=i^2 ; j++)
    {
      for( k=1 ; k<=n/2 ; k++)
      {
        print (BHU);
      }
    }
  }
}
```

$i = 1$	$i = 2$	$i = 3$	…	$i = n$
$j = 1$ time	$j = 4$ times	$j = 9$ ti.		$j = n^2$
$k = \frac{n}{2} * 1$	$k = n/2 * 4$	$k = n/2 * 9$		$k = n/2 * n^2$

$$\frac{n}{2}\left(1 + 4 + 9 \cdots + n^2\right)$$

$$= \frac{n}{2}\left(\frac{n(n+1)(2n+1)}{6}\right)$$

$$= O(n^4)$$

```
A()
{
  for(i=1; i<n; i=i*2)
      print(BHU);
}
```

1 2 4 8 . . . . n

2^0 2^1 2^2 2^3 . . . 2^K

$2^K = n$

$K = \log n$

$O(\log_2 n)$

```
A()
{
  int i, j, k;
  for( i=n/2; i<=n; i++)          --> n/2
     for (j=1; j<=n/2; j++)       --> n/2
        for(k=1; k<=n; k=k*2)     --> log2 n
           print(BHU)
        }
     }
  }
}
```

$\left(\frac{n}{2} * \frac{n}{2} * \log_2 n\right)$

$\Rightarrow \underline{\underline{O(n^2 \log_2 n)}}$

assume $n \geq 2$

```
A ()
{
  while (n > 1)
  {
    n = n/2
  }
}
```

$n = 2^k$

$k = \log_2 n$

$n = 20$

$\lfloor \log_2 20 \rfloor$

$O(\log_2 n)$

```
A ()
{
  for (i=1 ; i<=n ; i++)
    for (j=1 ; j<=n ; j=j+i)
      print (B A U)
  }
}
```

$i=1$	$i=2$	$i=3$	...	$i=k$
$j=1$ to n	$j=1$ to n	$j=1$ to n		$j=1$ to n
n times	$n/2$ times	$n/3$ times		n/k

$i = n$

$j = 1$ to n

n/n

$$n\left(1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}\right)$$

$$= O(n \log n)$$

Recursive Function

```
A(n)
{
  if ( )
  return (A(n/2) + A(n/2))
}
```

$$\boxed{T(n) = c + 2T(n/2)}$$

$$T(n) = 1 + T(n-1) \quad \text{when } n > 1$$
$$= 1 \quad \text{when } n = 1$$

```
A(n)
{
  if (n>1)
  return (A(n-1))
}
```

$$\boxed{T(n) = 1 + T(n-1)} \quad -① $$

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

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

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

$$= O(n)$$

Back substitution →

$$T(n-1) = 1 + T(n-2) \quad -②$$

$$T(n-2) = 1 + T(n-3) \quad -③$$

$$T(n) = n + T(n-1) \quad , \quad n > 1$$
$$= 1 \quad , \quad n = 1$$

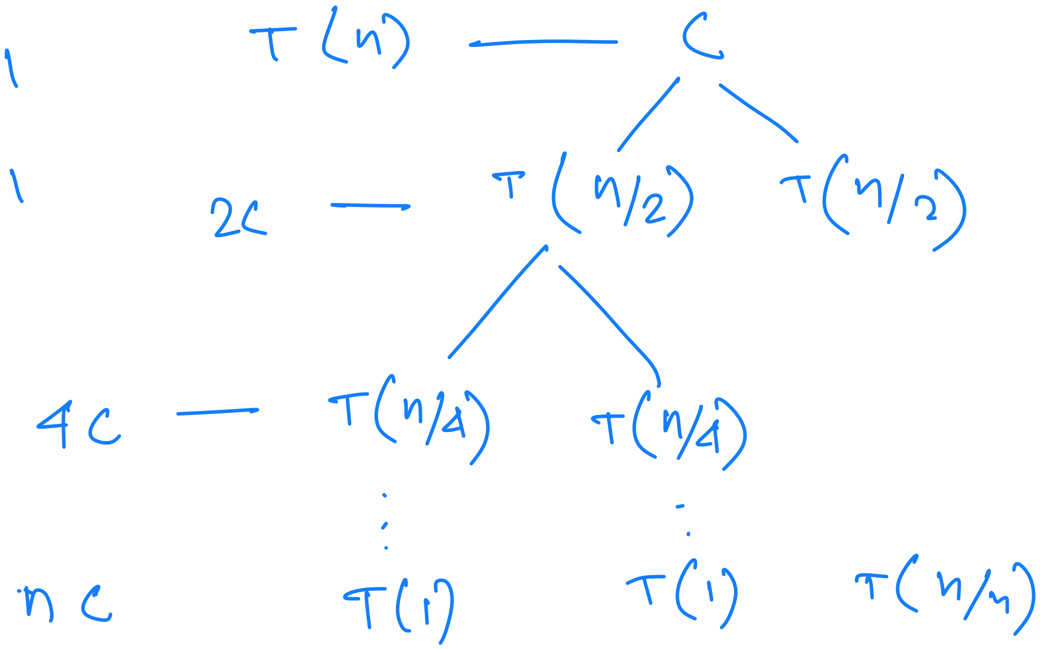
$$T(n-1) = (n-1) + T(n-2)$$
$$T(n-2) = (n-2) + T(n-3)$$

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

$$T(n) = n + T(n-1)$$
$$= n + (n-1) + T(n-2)$$
$$= n + (n-1) + (n-2) + T(n-3)$$
$$\vdots$$
$$= n + (n-1) + (n-2) \cdots (n-k) + T\underbrace{[n-(k+1)]}_{1}$$
$$= n + (n-1) + (n-2) \cdots 2 + 1$$
$$= \frac{n(n+1)}{2} = \underline{\underline{O(n^2)}}$$

Recursion Tree Method

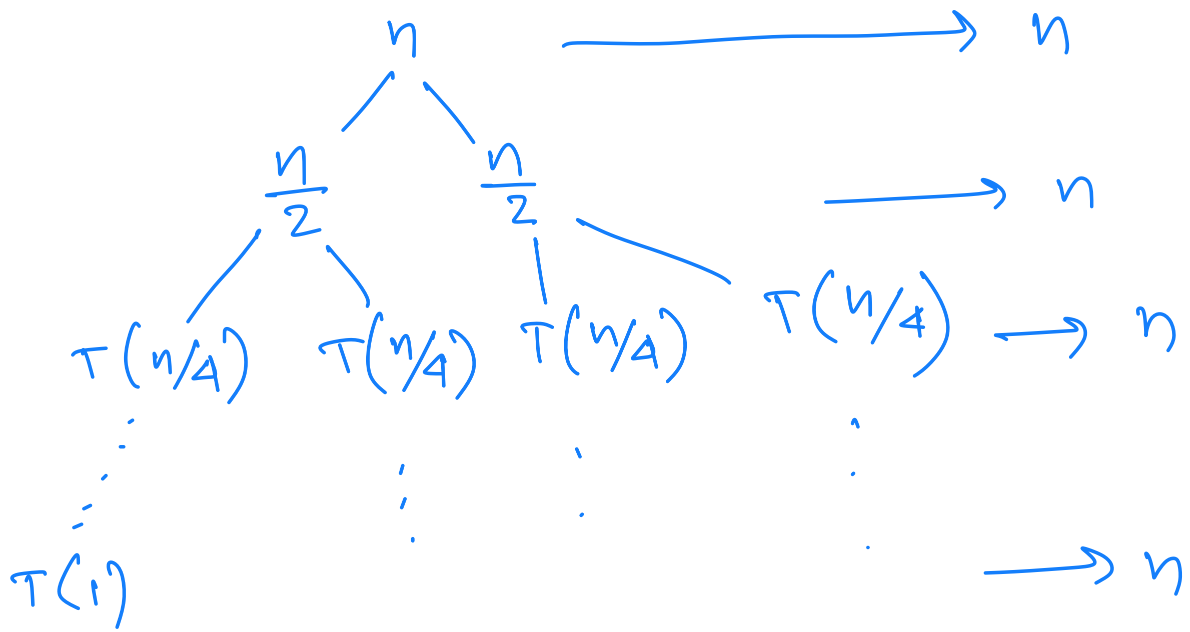
$$T(n) = 2T(n/2) + c \quad , n > 1$$
$$= c \quad ; n = 1$$



$$c + 2c + 4c + 8c + \cdots + nc$$
$$= c(1 + 2 + 4 \cdots + n) \quad \text{assume}$$
$$= c(2^0 + 2^1 + 2^2 + \cdots + 2^k) \quad n = 2^k$$
$$= c\left[\frac{1(2^{k+1} - 1)}{(2-1)}\right]$$
$$= c(2^{k+1} - 1)$$
$$= c(2n - 1)$$
$$= O(n)$$

$$T(n) = 2T(n/2) + n \quad , \quad n > 1$$
$$= 1 \quad , \quad n = 1$$

$$\frac{n}{2^0} + \frac{n}{2^1} + \frac{n}{2^2} + \dots \frac{n}{2^k}$$



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

$(k+1)$ levels

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

n $\qquad$ $(\log n)^{100}$

$\log n$ $\qquad$ $100 \log \log n$

let assume $n = 2^{128}$

$\log 2^{128}$

$= \underline{128}$

$100 \log \log 2^{128}$

$\Rightarrow 100 \log 128$

$\Rightarrow 100 * \log 2^{7}$

$\Rightarrow 100 * 7$

$= \underline{700}$ ✓

$n = 2^{1024}$

$\log 2^{1024}$

$= \underline{1024}$ ✓

$100(\log \log 2^{1024})$

$\Rightarrow 100 * \log 1024$

$\Rightarrow 100 * \log 2^{10}$

$\Rightarrow 100 * 10$

$\Rightarrow \underline{1000}$

$n^{\log n}$ $\qquad$ $n \log n$

$\log n \log n$ $\qquad$ $\log n + \log \log n$

assume $n = 2^{1024}$

$\underline{1024 * 1024}$ ✓

$1024 + \log \log n$

$\Rightarrow 1024 + \log \log 2^{1024}$

$\Rightarrow 1024 + \log 1024$

$\Rightarrow 1024 + \log 2^{10}$

$\Rightarrow 1024 + 10$

$\Rightarrow \underline{1034}$

$$f(n) = \begin{cases} n^3 & 0 < n < 10\,000 \\ n^2 & n \geq 10\,000 \end{cases}$$

$$g(n) = \begin{cases} n & 0 < n < 100 \\ n^3 & n > 100 \end{cases}$$

	0 - 99	100 - 9999	
$f(n)$	n^3	n^3	n^2
$g(n)$	n	n^3	n^3

$f_1 = 2^n$

$f_2 = n^{3/2}$

$f_3 = n \log n$

$f_4 = n^{\log n}$

$$T(n) = aT(n/b) + \theta(n^k \log^p n)$$ $a \geq 1$, $b > 1$, $k \geq 0$ and p is real number

① if $a > b^k$, then $T(n) = \theta(n^{\log_b a})$

② if $a = b^k$

ⓐ if $p > -1$, then $T(n) = \theta(n^{\log_b a} \log^{p+1} n)$

ⓑ if $p = -1$, then $T(n) = \theta(n^{\log_b a} \log \log n)$

ⓒ if $p < -1$, then $T(n) = \theta(n^{\log_b a})$

③ if $a < b^k$

ⓐ if $p \geq 0$, then $T(n) = \theta(n^k \log^p n)$

ⓑ if $p < 0$, then $T(n) = \theta(n^k)$

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

$a = 3, \ b = 2, \ k = 2, \ p = 0$

$a < b^k$

3 a $T(n) = \Theta(n^2 \log^0 n)$

$= \Theta(n^2)$

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

$a = 4, \ b = 2, \ k = 2, \ p = 0$

$4 = 2^2$

2 a $T(n) = \Theta(n^{\log_2 4} \log n)$

$= \Theta(n^2 \log n)$

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

$a=1,\ b=2,\ k=2,\ p=0$

$1 < 2^2$

3) (a) $T(n) = \Theta(n^2 \log^0 n)$

$= \Theta(n^2)$

$$T(n) = 16\,T(n/4) + n$$

$a=16,\ b=4,\ k=1,\ p=0$

$16 > 4$

$T(n) = \Theta(n^{\log_b a})$

$= \Theta(n^{\log_4 16})$

$= \underline{\underline{\Theta(n^2)}}$

$$T(n) = 2^n\,T(n/2) + n^n \quad \times$$

$$T(n) = 2T(n/2) + n\log n$$

$a=2,\ b=2,\ k=1,\ p=1$

2) (a) $T(n) = \Theta(n^{\log_b a} \log^{p+1} n)$

$= \Theta(n \log^2 n)$

$= \underline{\Theta(n \log^2 n)}$

$$T(n) = \sqrt{2}\,T(n/2) + \log n$$

$a=\sqrt{2},\ b=2,\ k=0,\ p=1$

$\sqrt{2} > 2^0$

$T(n) = \Theta(n^{\log_b a}) = \Theta(n^{\log_2 \sqrt{2}})$

$= \Theta(\sqrt{n})$

Insertion_sort(A)
{
for j = 2 to A.length
key = A[j]
i = j-1
while (i > 0 and A[i] > key
A[i+1] = A[i]
i = i-1

A[i+1] = key
}

$\boxed{\text{Inplace}}$

		comparison		movement		
when	j = 2	1	+	1	= 2	worst case
	j = 3	2	+	2	= 4	
	j = 4	3	+	3	= 6	
	⋮					
	j = n	(n-1)	+	(n-1)	= 2(n-1)	

$$2+4+6+\cdots+2(n-1)$$
$$= \cancel{2}\,\frac{n(n-1)}{\cancel{2}}$$
$$= n^2 - n$$
$$= O(n^2)$$

In best case
$$\Omega(n-1)$$
$$= \underline{\underline{\Omega(n)}}$$

Merge(A, p, q, r)
{

$\boxed{\text{out of place}}$

$n_1 = q - p + 1$

$n_2 = r - q$

Let L[1 .... n_1+1] and R[1 .... n_2+1] be new arrays

for(i = 1 to n_1)

L[i] = A[p + i - 1]

for (i = 1 to n_2)

R[j] = A[q + j]

copy this to new list → n

L[n_1 + 1] = ∞

R[n_2 + 1] = ∞

i = 1 , j = 1

for(k = p to r)

if (L[i] ≤ R[j])

A[k] = L[i]

i = i + 1

else A[k] = R[j] ; j = j + 1 ;

n comparison and n copies to new array (n + n)

```
merge_sort (A, p, r)
{
    if p < r
        q = ⌊(p + r)/2⌋
        merge_sort (A, p, r)
        merge_sort (A, q+1, r)
        merge (A, p, q, r)
}
```

Merge (A, p, q, r)
{

$n_1 = q - p + 1$

$n_2 = r - q$

Let L[1 .... n_1+1] and R[1 .... n_2+1] be new arrays

$\boxed{\text{out of place}}$

```
for( i = 1 to n1)
     L[i] = A[p + i - 1]
for ( i = 1 to n2)
     R[j] = A[q + j]
```

copy this to new list → n

```
L [n1 + 1] = ∞
R[n2 + 1] = ∞

i=1 , j = 1
for ( k= p to r)
   if ( L[i] ≤ R[j])
        A[k] = L[i]
        i = i + 1
   else  A[k] = R[j] ; j = j + 1;
```

n comparison and n copies to new array (n + n)

Example of Merge Sort

1	2	3	4	5	6	7	8
1	5	7	8	2	4	6	9

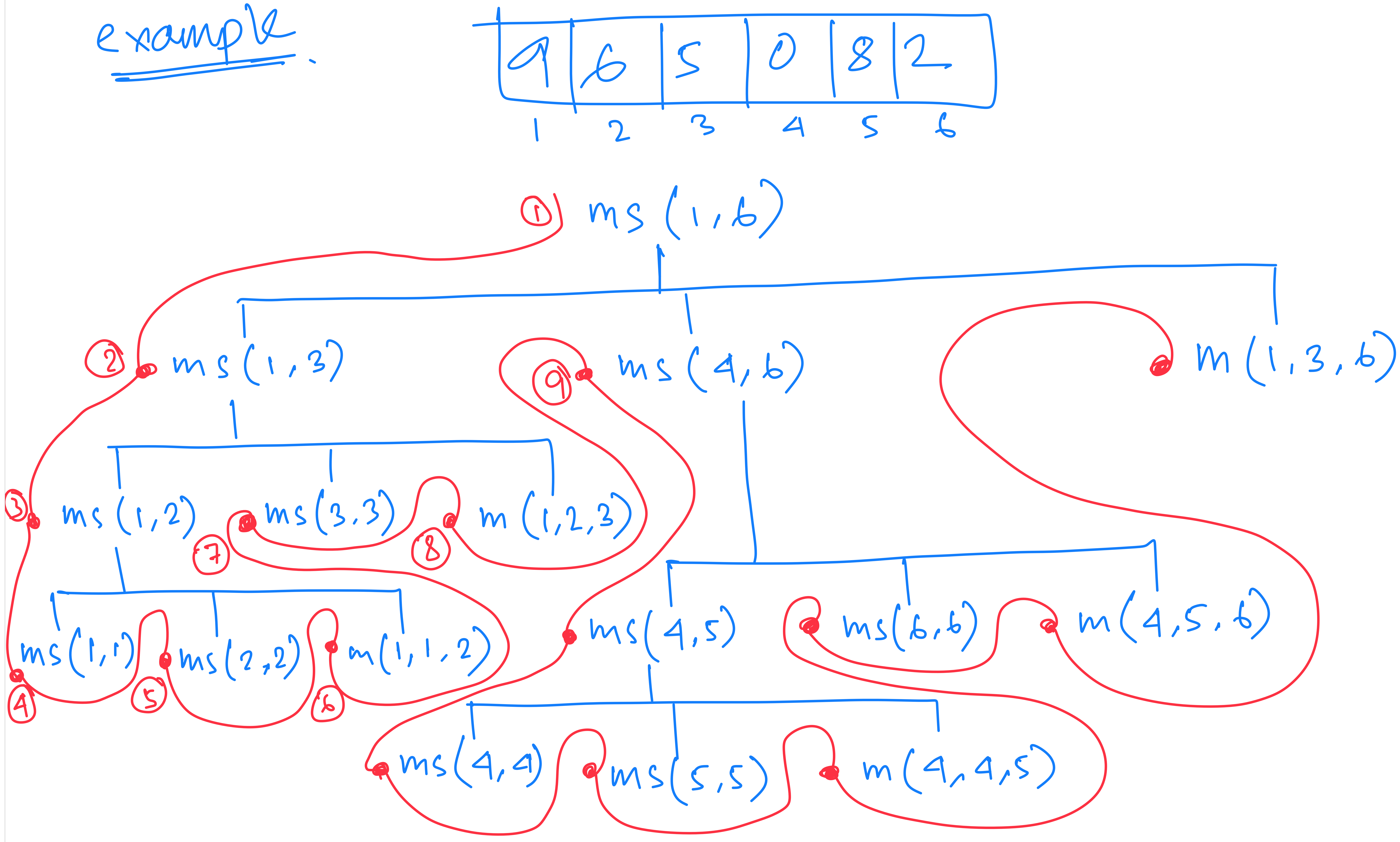
1	5	7	8	∞

2	4	6	9	∞

to copy n elements, it will take $O(n)$

```
merge_sort(A, p, r)
{
    if p < r
        q = ⌊(p+r)/2⌋
        merge_sort(A, p, r)
        merge_sort(A, q+1, r)
        merge(A, p, q, r)
}
```

example



9 6 5 0 8 2
1 2 3 4 5 6
1 ms(1,6)
2 ms(1,3)
9 ms(4,6)
m(1,3,6)
3 ms(1,2)
7 ms(3,3)
8 m(1,2,3)
4 ms(1,1)
5 ms(2,2)
6 m(1,1,2)
ms(4,5)
ms(6,6)
m(4,5,6)
ms(4,4)
ms(5,5)
m(4,4,5)

Quick Sort

```
Partition(A, p, r)
{
    x = A[r]
    i = p - 1
    for ( j = p to r - 1)
    {
        if (A[j] <= x)
        {
            i = i + 1
            exchange A[i] with A[j]
        }
    }
    exchange A[i+1] with A[r]
    return i+1
}
```

9	6	5	0	8	2	4	7

```
Quick_sort(A, p, r)
{
  if (p<r)
  {
    q = partition(A, p, r)
    Quick_sort(A, p, q-1)
    Quick_sort(A, q+1, r)
  }
}
```