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

$= \frac{n}{2} + n - 2$

$\boxed{T(n) = \frac{3n}{2} - 2}$

$2(n-1) - 2n - 2 > \frac{3n}{2} - 2$

21/1/23

① for (i=0; i<n; i++){

statement

}

$O(n)$

② for (i=n; i>0; i--){

statement

}

$O(n)$

③ for (i=0; i<n; i=i+2){

$n/2$

st

}

$O(n)$

④ for (i=1; i<n; i=i*2){

st

}

$O(\log n)$

⑤ while ( ){

st

}

if st 0 times while check 1 time

$O(n)$

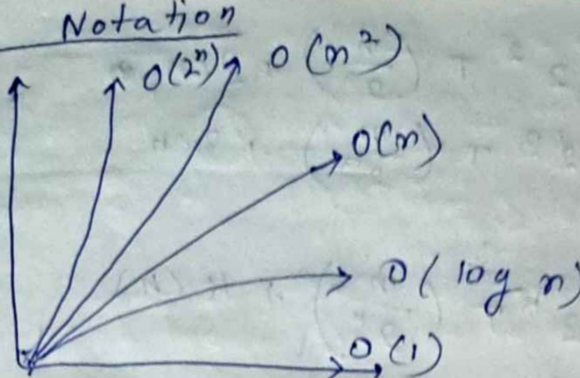
⑥ do {

st

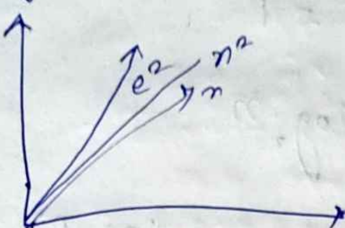
$O(n)$

} while ( );

Asymptotic Notation



$1 < \log n < \sqrt{n} < n \log n < n^2 < n^2 \dots n^n < 2^n < 3^n \dots n$



① Upper boundary O
② Lower boundary Ω
③ Average boundary Θ

22/1/25

Quick Sort

$\xleftarrow{n/2}$ (P) $\xrightarrow{n/2 - 1}$

Pivot
↓
left < $\boxed{P}$ < Right

```
while (A[i] < A[P]) { i++ }
while ( A[j] > A[P] ) { j-- }
    if (i < j) { swap( A[i], A[j] ) }
```

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

↑ for considering all element as Pivot

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

$\boxed{T(n) = 2\,T\left(\frac{n}{2}\right) + cn}$, $n > 1$ replacing $T\left(\frac{n}{2}\right)$ with $n = \frac{n}{2}$

$= 2\left[2\,T\left(\frac{n}{2 \cdot 2}\right) + c\frac{n}{2}\right] + cn$

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

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

$$\vdots$$

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

$\boxed{T(n) = 1}$ when $n=1$ base condition/ terminating "

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

$\Rightarrow k = \log_2 n$

$\Rightarrow nT(1) + cn\log n$

$\Rightarrow n \cdot 1 + c \cdot n \log n$

$\Rightarrow n + cn\log n$

$\Rightarrow O(n\log n)$

28/1

Asymptotic Notation

① Big - oh

① If $f(n) = O(g(n))$ where c_0 and n is +ve constant if exists $n \geq n_0$

$f(n) \leq c * g(n)$

Q. $f(n) = 2n^3 + 5n + 3$ find upper boundary and lower boundary

Quick Sort

Jenny

key

	Pivot	

value < Pivot   Value > pivot

eg: Arr

0	1	2	3	4	5	6
10	15	1	2	9	16	11

let pivot = 10 , Find proper place of pivot

2, 1, 9	10	15, 11, 16

now pivot is at appropiate place

now sort, 0 – 2
4 – 6

eg

left (lower bound)   right (upper band)

7	6	10	5	9	2	1	15	7

start   start   end

pivot $\boxed{7}$

pivot

① start, 7 ≤ 7 $\boxed{✓}$
⇒ start ++

② start, 6 ≤ 7 $\boxed{✓}$
⇒ start ++

③ start, 10 ≤ 7 $\boxed{X}$
⇒ stop

pivot

① end, 7 > 7 $\boxed{X}$
⇒ stop

now swap start and end

⇒

7	6	7	5	9	2	1	15	10

start   start   end   end

① start, 7 ≤ 7 $\boxed{✓}$
start ++

② start, 5 ≤ 7 $\boxed{✓}$
start ++

③ start 9 ≤ 7 $\boxed{X}$
stop

① end, 10 > 7 $\boxed{✓}$
end --

② end, 15 > 7 $\boxed{✓}$
end --

③ end, 1 > 7 $\boxed{X}$
stop

now swap start, end

⇒

7	6	7	5	1	2	9	15	10

start end start end

① 1 ≤ 7 $\boxed{✓}$
② 2 ≤ 7 $\boxed{✓}$
③ 9 ≤ 7 $\boxed{X}$

① 9 > 7 $\boxed{✓}$
② 2 > 7 $\boxed{X}$

now start > end so no swap

now swap pivot and end

⇒

2	6	7	5	1	7	9	15	10

≤ pivot   > pivot

do same for left & right partition.

* partition exchange sort

Algo

```
partition (arr, lb, ub) {
    pivot = arr[lb]
    start = lb
    end = ub
  while (start < end) {
    while (a[start] <= pivot)
        start ++
    while (a[end] > pivot)
        end --
    if (start < end)
        swap (a[start], a[end])
  }
  swap (a[lb], a[end])
  return end;  ⟶ right position of pivot
}

Quick Sort (arr, lb, up) {
  if (lb < up) {
    loc = partition (A, lb, up);
    Quick sort (arr, lb, loc - 1);
    QuickSort( arr, loc + 1, ub);
  }
}
```

worst case $O(n^2)$

best case $O(n \log n)$

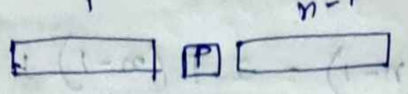
avg " "

worst case

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

$T(n+1) + O(n) \;; n > 1 \implies O(n^2)$

Average Case :

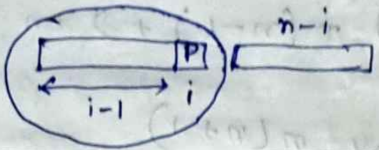


Pivot not at edge

$$T(n) = \frac{1}{n}\sum_{i=1}^{n-1} T(i) + T(n-i) + O(n) \qquad n > 1$$

conquer

$O(n^2)$



$$n\,T(n) = \sum_{i=1}^{n-1}\left[T(i) + T(n-i)\right] + cn^2 \quad \text{----} \quad (1)$$

put $n = n-1$

$$(n-1)\,T(n-1) = \sum_{i=1}^{n-2}\left[T(i) + T(n-1-i)\right] + c(n-1)^2 \quad \text{----} \quad (2)$$

(1) - (2) $\qquad [1, 2 \ldots n-2]\ n-1$

$$n\,T(n) - (n-1)\,T(n-1) = T(n-1) + cn^2 - c(n-1)^2$$

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

$$\Rightarrow T(n) = \frac{1}{n}\sum_{i=1}^{n-1} T(i) + T(n-i) + O(n)$$

$$\Rightarrow T(n) = \frac{1}{n}\sum_{i=1}^{n-1} 2T(i) + O(n)$$

$$\Rightarrow n\,T(n) = \sum_{i=1}^{n-1} 2T(i) + cn^2$$

$$\Rightarrow n\,T(n) = 2\sum_{i=1}^{n-1} T(i) + cn^2 \quad \text{———} \quad (1)$$

$[T(1) + T(2) \ldots + T(n-2) + T(n-1)]$

put, $n = n-1$

$$(n-1)\,T(n-1) = 2\sum_{i=1}^{n-2} T(i) + c(n-1)^2 \quad \text{———} \quad (2)$$

$[T(1) + T(2) \ldots T(n-2)]$

① - ②

$n\,T(n) - (n-1)\,T(n-1) = 2\,T(n-1) + cn^2 - c(n-1)^2$

$n\,T(n) = (n+1)\,T(n-1) + c(n+n-1)(n-n+1)$

$n\,T(n) = (n+1)\,T(n-1) + 2cn$

Divide $n-1$ by $n(n+1)$

$\cancel{\frac{T(n)}{(n+1)} = \frac{T(n-2)}{n-1} + \frac{2c}{n} + }$?

?

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

putting $n=n-1$ in $T(n)$ and substitute. $T(n-1)$

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

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

if $n=2$, (Terminating condition)

$= \frac{T(1)}{2} + \frac{2c}{3} + \dots + \frac{2c}{n+1}$

$\frac{T(n)}{(n+1)} = \frac{1}{2} + \frac{2c}{3} + \dots + \frac{2c}{n+1}$

$= \frac{1}{2} + 2c\left[\frac{1}{3} + \dots + \frac{1}{n+1}\right]$

$= \frac{1}{2} + 2c\left[\frac{1}{1} + \frac{1}{2} + \frac{1}{3} \dots \frac{1}{n+1}\right] - 3c$

$= \frac{1}{2} + 2c\log(n+1) - 3c$

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

$= 2cn\log n$

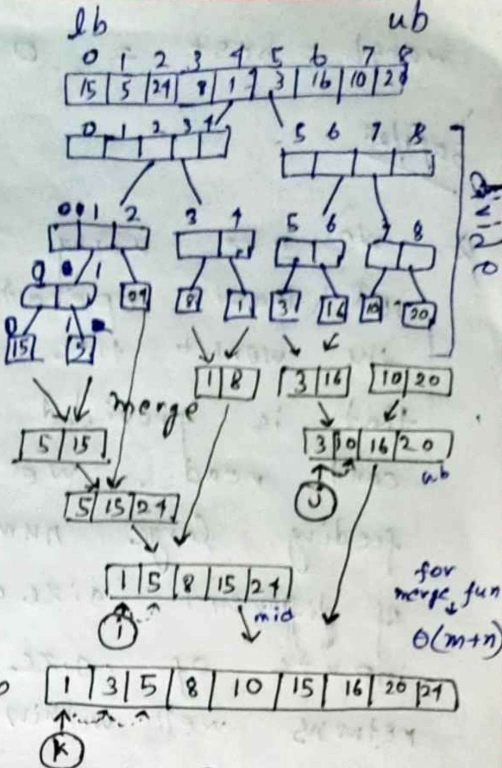
$\boxed{O(n\log n)}$

Merge Sort

Jenny

① divide list till get 1 element

```
[Mergesort] (A, lb, ub){
    if (lb < ub) {
        mid = (lb + ub)/2
    MergeSort (A, lb, mid);
    MergeSort (A, mid+1, Ub);
    merge(A, lb, mid, ub);
    }
}
[merge] (A, lb, mid, ub) {
    i = lb;
    j = mid + 1;
    k = lb
    while ( i <= mid && j <= ub) {
        if (a[i] <= a[j]) {
            b[k] = a[i]
            i++;
        }
        else {
            b[k] = a[j]
            j++
        }
        k++
    }
    if ( i > mid) {
        while (j <= ub) {
            b[k] = a[j]; j++; k++;
        }
    }
    else {
        while( i <= mid){
            b[k] = a[i]
            i++; j++;
        }
    }
    for (k = lb ; k <= ub; k++){
        a[k] = b[k]
    }
}
```



* left & right sub array, element wise check করব and যেটা ছোট b array তে রাখব

* if element of left is ছোট, copy it to b[k] and increment i

* i reached to end but j didnt reach end. ⇒ copy remainings of j in b array.

* j reached beyond end but remaining element in i ⇒ copy remainings of i in t b array.

* copy everything of b to a

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

30/1/25

2. we are using a computer that performs 10^8 basic operations per second. determine the worst time complexity of a library func that is provided to us. whose code we can't read. we test the function by feeding large numbers of random inputs of different sizes. We find that for inputs of size 50, the function always returns well within one second, for inputs of size 500 it sometimes takes a couple of seconds and for inputs of size 5000 it takes over 15 min. what is a resonable conclusion we can draw about the worst case time complexity of library function.

Merge Sort

SKM

0 56 9

$\leftarrow n_1 \rightarrow \leftarrow n_2 \rightarrow$

```
Merge (A, l, mid, r){
    n1 <- mid - l              i <- l
    n2 <- (r - (mid+1))        j <- mid+1
                               k <- 0
    while (k < n1+n2)
        if (A[i] < A[j] OR j == n2+n1)
            c[k++] = A[i];  i++;
        if ((A[i] > A[j] OR I == n1)
            c[k++] = A[j];  j++;
```

Time Complexity

$$T(n) = \underbrace{T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right)}_{\text{mergesort}} + \underbrace{O(n)}_{\text{merge}}$$

$\boxed{\text{Best} = \text{worst} = \text{Average}}$

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

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

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

$$= 2 \cdot 2\, T\, \frac{n}{2 \cdot 2} + 2 \cdot cn$$

$$\vdots$$

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

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

$$n = 2^k$$

$$k = \log n$$

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

$$T(n) = O(n \log n)$$ Best/worst/avg