CSA0666 - Design & Analysis of Algorithms 192311044
Analytical Assignment - 2 Girija.B

1. If $T_1(n) \in O(g_1(n))$ and $T_2(n) \in O(g_2(n))$, then $T_1(n)+T_2(n) \in O(\max\{g_1(n), g_2(n)\})$. Prove the assertions.

For any real numbers, a_1, b_1, a_2, b_2, such that $a_1 \leq b_1$ and $a_2 \leq b_2$, we have, $a_1+a_2 \leq 2 \max\{b_1, b_2\}$.

Given,

$T_1(n) \in O(g_1(n))$, for all non negative integers n_1 & c_1,

$T_1(n) \leq c_1 g_1(n)$ for all $n \geq n_1$

also, since $T_2(n) \in O(g_2(n))$, then, there is constant c_2 & non negative integer n_2.

$T_2(n) \leq c_2 g_2(n)$ for all $n \geq n_2$.

$c_3 = \max\{c_1, c_2\}$ & $n_0 = \max\{n_1, n_2\}$.

$$T_1(n)+T_2(n) \leq c_1 g(n) + c_2 g_2(n)$$
$$\leq c_3 g_1(n) + c_3 g_2(n)$$
$$= c_3\{g_1(n)+g_2(n)\}$$
$$\leq \max\{g_1(n)+g_2(n)\}$$

$\therefore T_1(n)+T_2(n) \in O(\max\{g_1(n)+g_2(n)\})$

Hence Proved.

2. Find time complexity of below recurrence equation.

3. a) $T(n) = \begin{cases} 2T(n/2)+1 & \text{if } n>1 \\ 1 & \text{otherwise} \end{cases}$

Given, $T(n) = \begin{cases} 2T(n/2)+1 & \text{if } n>1 \\ 1 & \text{otherwise} \end{cases}$

By master theorem,

$T(n) = aT(n/b)+f(n)$

$a=2 \; ; \; b=2; \; f(n)=1 \; ; \; K=0$

$\log_b^a = \log_2^2 = 1$

Since, $\log_b^a > K$, $T(n) = \theta(n \log_b^a)$

$= \theta(n')$

$= \theta(n)$.

$\therefore$ Time Complexity, $T(n) = \theta(n)$ //

4. $T(n) = \begin{cases} 2T(n-1) & \text{if } n>0 \\ 1 & \text{otherwise} \end{cases}$

Given, $T(n) = 2T(n-1)$ —① By Backward substitution,

Substitute, $n = n-1$

$T(n-1) = 2T(n-1-1) = 2T(n-2)$ —②

Substitute ② in ①

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

$= 2^2 T(n-2)$ —③

Substitute $n = n-2$

$T(n-2) = 2T(n-2-1) = 2T(n-3)$ —④

Substitute ④ in ③

$$T(n) = 2^2[2T(n-3)]$$

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

$$\vdots$$

$$T(n) = 2^K T(n-K)$$

Stop when, $n = K \Rightarrow n-K = 0$.

$$T(n) = 2^n T(0)$$

Given, $T(0) = 1$

So, $T(n) = 2^n \cdot 1$

$$= 2^n$$

$\therefore$ Time Complexity is $O(2^n)$ //

5. Big O notation: show that $f(n) = n^2+3n+5$ is $O(n^2)$.

Given, $f(n) = n^2+3n+5$.

Since, $O(n^2)$ needed to prove,

$$f(n) \leq c.n^2$$

$$f(n) = n^2+3n+5$$

$$n^2+3n+5 \leq c.n^2$$

for $n > 1$,

$$n^2+3n+5 \leq n^2+3n^2+5n^2$$

$$n^2+3n+5 \leq (1+3+5)n^2$$

$$n^2+3n+5 \leq 9n^2$$

now, $c = 9$, $n_0 = 1$.

$\therefore$ $f(n)$ is in $O(n^2)$ with $c = 9$.

6. Big Omega Notation: Prove that $g(n) = n^3 + 2n^2 + 4n$ is $\Omega(n^3)$

Given, $g(n) = n^3 + 2n^2 + 4n$.

$f(n) = n$

for Ω, condition

$f(n) \geq c \cdot g(n)$

when,

$n = 1$, $1 \geq 1 + 2 + 4$

$1 \geq 7$ - NO

$n = 2$, $2 \geq$

$n^4 = n^3 + 2n^2 + 4n$

when,

$3^4 = 81$

$81 = 27 + 18 + 12$

$81 = 57$

$f(n) = n^4$ with $n \geq 3$, $g(n)$ is $\Omega(n^3)$.

7. Big Theta Notation: Determine whether $h(n) = 4n^2 + 3n$ is $\theta(n^2)$ or not:

Given, $h(n) = 4n^2 + 3n$ is $\theta(n^2)$ or not.

for theta, condition is $c_1 g(n) \leq f(n) \leq c_2 g(n)$

so, we need to show,

$$c_1 n^2 \leq 4n^2 + 3n \leq c_2 n^2$$

Upper Bound: $O(n^2)$

$$4n^2 + 3n \leq c_2 n^2$$

for $n \geq 1$, $4n^2 + 3n \leq 4n^2 + 3n^2 = 7n^2$

$\therefore 4n^2 + 3n \leq 7n^2$

Lower Bound: (Ωn^2)

$4n^2+3n \geq c_1 n^2$ → for large n, $4n^2$ will dominate $3n$

$4n^2 + 3n \geq 4n^2$

$4n^2+3n \geq 4n^2$ ($c_1=4$)

$\therefore h(n)=4n^2+3n$ is $\theta(n^2)$ with $c_1=4$, $c_2=7$, $n_0=1$.

$\therefore h(n)=4n^2+3n \in \theta(n^2)$.

8. Let $f(n)=n^3-2n^2+n$ and $g(n)=n^2$, show whether $f(n)=\Omega(g(n))$ is true or false and justify your answer.

Given, $f(n)=n^3-2n^2+n$

$g(n)=n^2$

for $\Omega(g(n))$, $f(n) \geq c.g(n)$ $c=1$,

$n^3-2n^2+n \geq c.n^2$

$n=1$, $1-2+1 \geq c.1$ ×

$n=2$ $8-8+8 \geq 4$ ✓

$8 \geq 4$

So, for $n \geq 2$, $f(n)=\Omega(g(n))$ is true since it satisfies the Big Omega Condition.

9. Determine whether $h(n)=n\log n+n$ is in $\theta(n\log n)$ prove a rigorous proof for your conclusion.

We need to ~~Prove~~ show that, there exist a positive constant c_1, c_2 and n_0 such that for all $n \geq n_0$,

$C_1 \cdot n\log n \leq h(n) \leq C_2 \cdot n\log n$.

Upper Bound:

$h(n) = n\log n + n$.

for $n \geq 1$, $\log n$ is Positive,

$\therefore n\log n + n \leq n\log n + n\log n$

$n\log n + n \leq 2n\log n$.

$\therefore C_2 = 2$,

$h(n) = n\log n + n \leq 2n\log n$. -Proved.

Lower Bound:

for $n \geq 1$, $\log n$ is Positive,

$n + n\log n \geq n\log n$

$\therefore C_1 = 1$; $h(n) = n\log n + n \geq n\log n$.

Thus, $n\log n \leq n\log n + n \leq 2n\log n$, for all $n \geq 1$.

$\therefore$ We Prove the notation of Big theta with values, $C_1 = 1$, $C_2 = 2$ & $n_0 = 1$.

Hence,

$h(n) = n\log n + n$ is in $\theta(n\log n)$.

10 Solve the following recurrence relations and find the order of growth for solutions

$T(n) = 4T(n/2) + n^2 \quad T(1) = 1$.

$T(n) = aT + HT(n/2) + n^2 \quad T(1) = 1.$

now, master Theorem,

$$T(n) = aT(n/b) + f(n)$$

here,

$a = 4,\ b = 2,\ f(n) = n^2,\ K = 2,\ P = 1.$

now,

$$\log_b^a = \log_2^4 = 2.$$

$\log_b^a = K$, P Compare P,

~~P~~ so,

$P \geq -1$, so, $\theta(n^K \log_n^{P+1})$

$\Rightarrow \theta(n^2 \log_n^1)$

$\Rightarrow \theta(n^2 \log n).$

$\therefore$ Order of growth of $T(n) = 4T(n/2) + n^2$ is $\theta(n^2 \log n)$ //.

11. Given an array of [4, -2, 5, 3, 10, -5, 2, 8, -3, 6, 7, -4, 1, 9, -1, 0, -6, -8, 11, -9] integers. find max and min product that can be obtained by multiplying two integers from array.

Given array:

[4, -2, 5, 3, 10, -5, 2, 8, -3, 6, 7, -4, 1, 9, -1, 0, -6, -8, 11, -9].

Sorting the array:

[-9, -8, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]

Max Product:

↳ Product of two largest (+ve) numbers

↳ Product of two Smallest (negative) numbers.

min. Product:

↳ Product of two largest Positive numbers (if one is 0 / negative).

↳ Product of two smallest (most negative numbers).

↳ Product of smallest negative number and largest +ve.

@SOL

max Product

↳ $10 \times 11 = 110$

↳ $(-9) \times (-8) = 72$

∴ max Product = 110.

min Product.

↳ $10 \times 11 = 110$

↳ $-9 \times -8 = 72$

↳ $-9 \times 11 = -99$

↳ second Smallest -ve & second largest Positive, $= -8 \times 10 = -80$

∴ min Product = -99.

∴ max product = 110, min product = -99.

Code:

```
function max_min_product (arr)
 arr.sort().
 max_product = max (arr[0]*arr[1], arr[-1]*arr[-2])
 min_product = min (arr[0]* arr[1], arr[0]* arr[-1],
                                    arr[1] * arr[-1]).
  return max_min product
max_product, min_product = fn max_min product (arr).
```

12. Demonstrate Binary Search method to search Key = 23 from array arr[] = {2, 5, 8, 12, 16, 23, 38, 56, 72, 91}

Given,

arr[] = {2(0), 5(1), 8(2), 12(3), 16(4), 23(5), 38(6), 56(7), 72(8), 91(9)}

⇒ low = 0
high = 9
mid = $\frac{0+9}{2}$
= 4.

arr[mid] = arr[4] = 16.

⇒ 16 < 23, search in right half. } 1st iteration.
now, low = mid + 1 = 5.

⇒ mid = 5+9/2 = 7 arr[mid] = arr[7] = 56. } 2nd iteration
now, 56 > 23 search in left half:
high = mid − 1 = 6.

⇒ mid = 5 + 6/2 = 5 arr[mid] = arr[5] = 23 } 3rd iteration
∴ 23 = 23, key found at index 5.

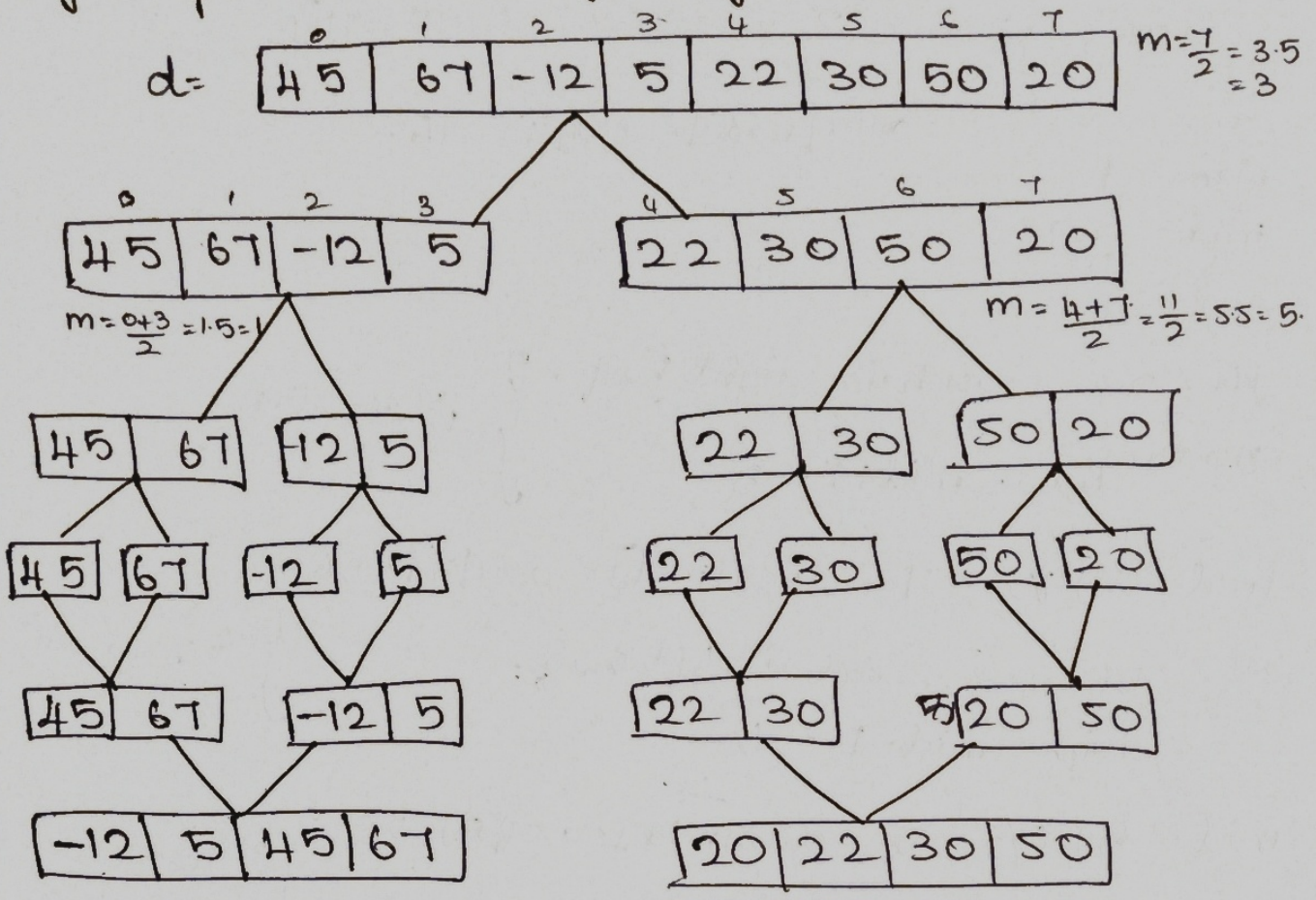
→ 2(0) 5(1) 8(2) 12(3) 16(4) | 23(5) 38(6) 56(7) 72(8) 91(9)
low mid high

→ 2 5 8 12 16 | 23 38 56 | 72 91
low mid high

→ 2 5 8 12 16 | 23 38 56 72 91
low high
mid

23 = 23,
found at index 5.

13. Apply merge sort and order the list, d = (45, 67, -12, 5, 22, 30, 50, 20). Set up a recurrence relation for no. of key comparisons made by mergesort.



Ordered list d = | -12 | 5 | 20 | 22 | 30 | 45 | 50 | 67 |

Recurrence relation:

Let no. of key comparison for list of size n be T(n).

for, n=1 no comparisons.

$T(1)=0$.

for $n>1$, merge sort divides list to two halves of $\frac{n}{2}$ each, each half involves $T(n/2)$. so, $2T(n/2)$.

Again merging takes at most n comparisons.

So recurrence relation, $T(n)=2T(n/2)+n$.

Solving,

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

master theorem, $T(n) = aT(n/b) + f(n)$.

$a = 2, b = 2, f(n) = n, K = 1$

$\log_b^a = \log_2^2 = 1.$

$\log_b^a = K$, so, P comparison.

$P = 0, 0 > -1$

$= \theta(n^K \log_n^{P+1})$

$= \theta(n^1 \log^1 n)$

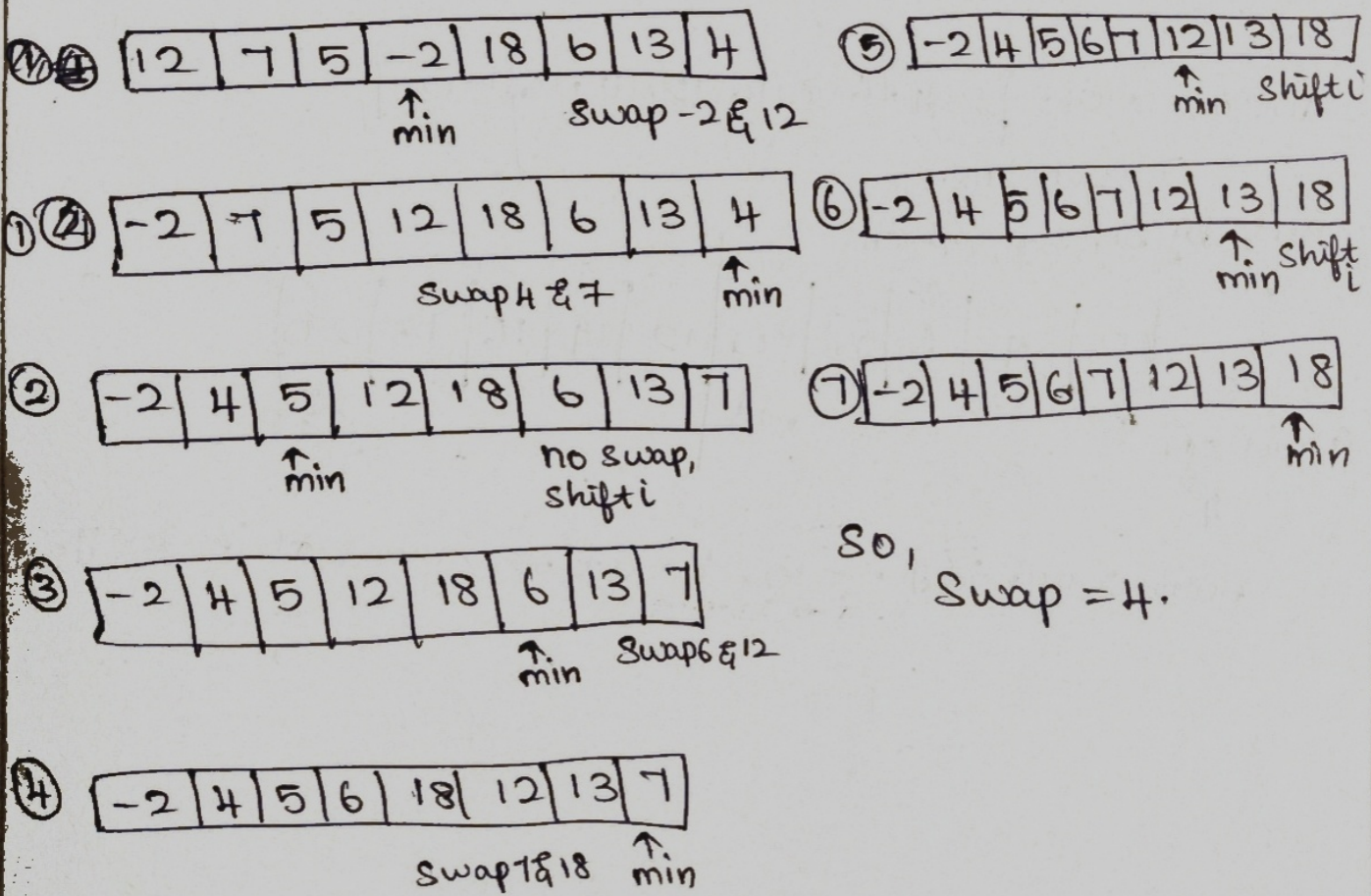
$= \theta(n \log n)$

⇒ Merge sort is have $O(n \log n)$ complexity.

14. Find no. of times to perform swapping for selection sort. Also estimate time complexity for order of notation set

S(12, 7, 5, -2, 18, 6, 13, 4).

S = 12, 7, 5, -2, 18, 6, 13, 4



So, Swap = 4.

Time Complexity:

```
arr: array of items
n: size
for (i=0; i<n-1; i++)          ---- n
min=i
    for (j=i+1; j<n; j++)      ---- n×n
        if arr[j] < arr[min] then --|
            min=j              --|
        end if                 --|
    end for                    --|
    if (min != i) then         --|
        swap arr[min] and arr[i]
    end if
end for
```

$T(n) = \theta(n^2)$.

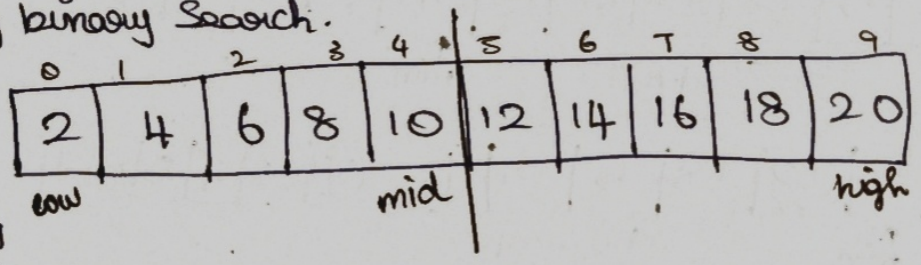
Swaps = 4.

15. Find index value of target value 10 using binary search from following list of elements [2, 4, 6, 8, 10, 12, 14, 16, 18, 20].

Given, arr = [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]

Target = 10.

now, by binary search.



(1) mid = $\frac{0+9}{2}$ = 4

now, arr[mid] == 10, target is found at index 4.

Algorithm:

low = 0

high = 9.

while low ≤ high

mid = low + high / 2

if arr[mid] == target

Print "target is found at index mid".

if arr[mid] < target

low = mid + 1

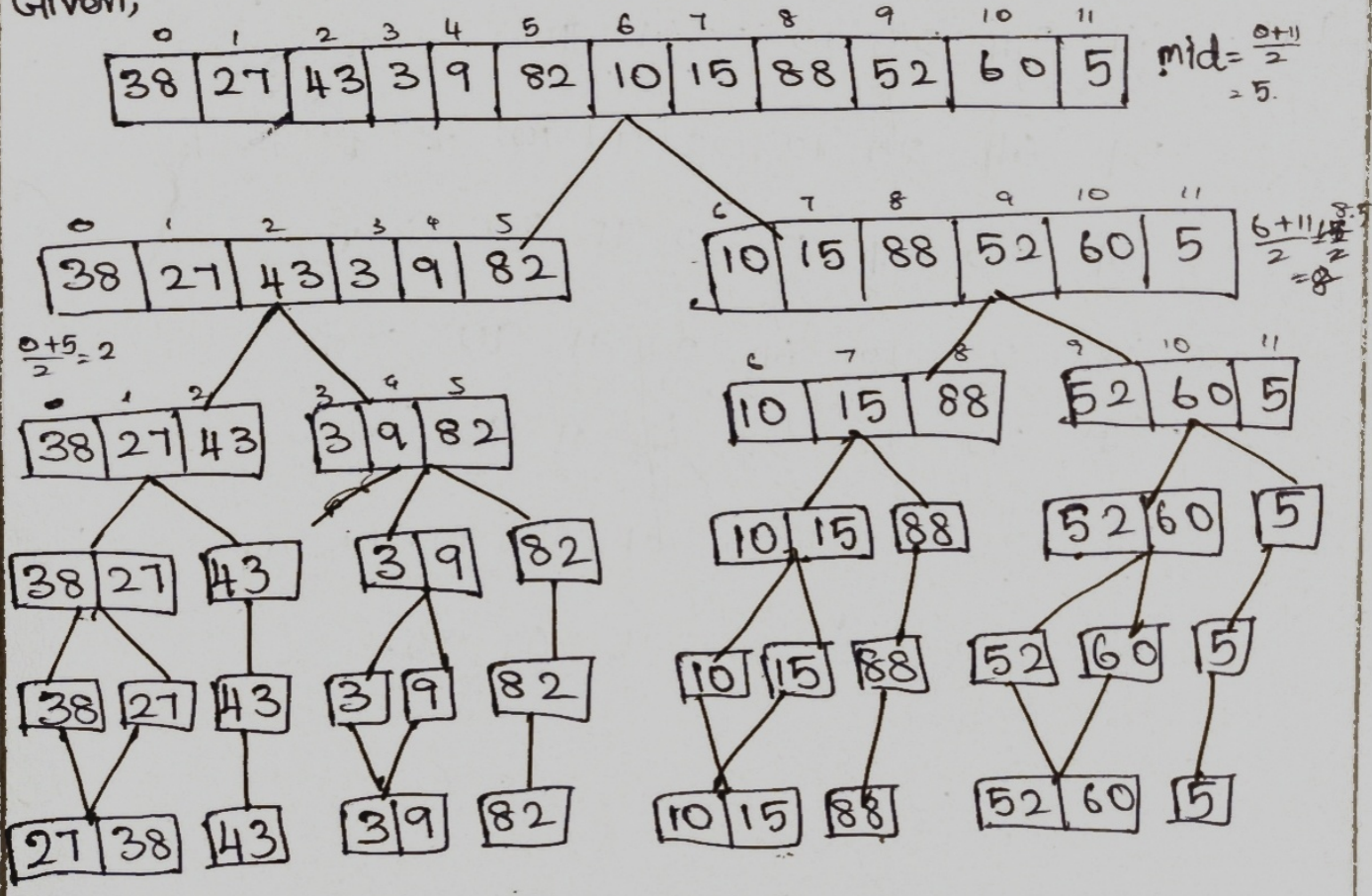
if arr[mid] > target

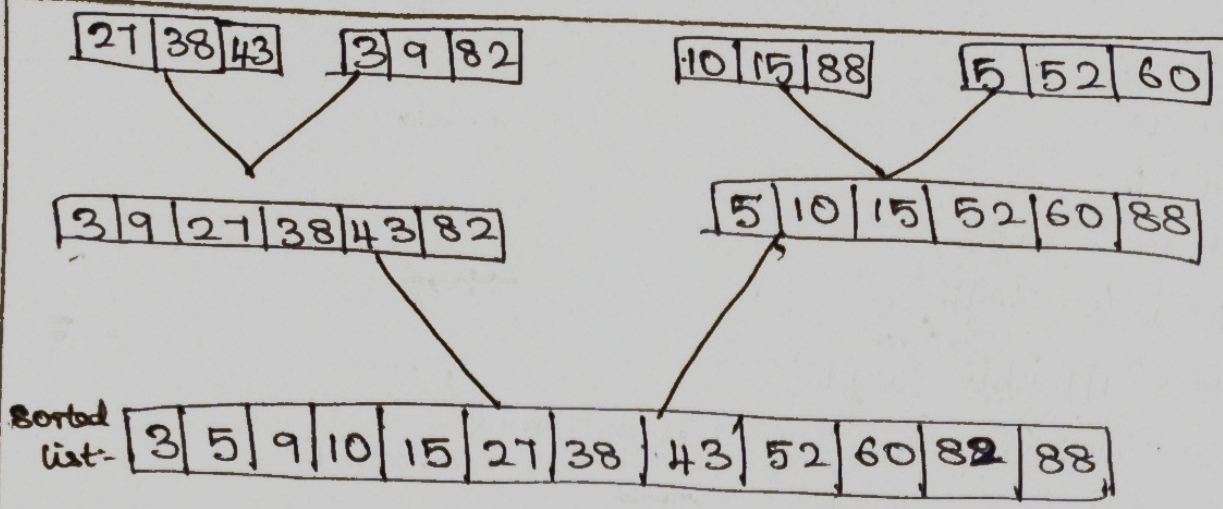
high = mid - 1

return -1.

16. Sort the following elements using merge sort divide & Conquere strategy [38, 27, 43, 3, 9, 82, 10, 15, 88, 52, 60, 5] and analyze complexity of algorithm.

Given,





Time Complexity:

Divide of list to two halves ⇒ log n times ⇒ n/2.

merging sorted two lists ⇒ n

∴ $O(n \log n)$

17. Sort array 64, 34, 25, 12, 22, 11, 90 using Bubble sort. What is time Complexity Bubble sort in best, worst & average Cases?

I1: 64 34 25 12 22 11 90 - swap

34 64 25 12 22 11 90 - swap

34 25 64 12 22 11 90 - swap

34 25 12 64 22 11 90 - swap

34 25 12 22 64 11 90 - swap

34 25 12 22 11 64 90 - no swap

~~34 25 12 22 11 64 90.~~

I2:

34 25 12 22 11 64 90 - swap

25 34 12 22 11 64 90 - swap

25 12 34 22 11 64 90 - swap

25 12 22 34 11 64 90 - swap

25 12 22 11 34 64 90 - no swap

25 12 22 11 34 64 90 - no swap

I3:

25 12 22 11 34 64 90 - swap

12 25 22 11 34 64 90 - swap

12 22 25 11 34 64 90 - swap

12 22 11 25 34 64 90 - no swap

12 22 11 25 34 64 90 - no swap

12 22 11 25 34 64 90 - no swap

I4:

12 22 11 25 34 64 90 - no swap

12 22 11 25 34 64 90 - swap

12 11 22 25 34 64 90 - no swap

12 11 22 25 34 64 90 - no swap

12 11 22 25 34 64 90 - no swap

12 11 22 25 34 64 90 - no swap

I5:

12 11 22 25 34 64 90 - swap

11 12 22 25 34 64 90 - no swap

11 12 22 25 34 64 90 - no swap

11 12 22 25 34 64 90 - no swap

11 12 22 25 34 64 90 - no swap

11 12 22 25 34 64 90 - no swap

Sorted array: [11, 12, 22, 25, 34, 64, 90].

Time Complexity:

Best Case:

↳ when array is already sorted, it occurs.

↳ So, it goes n-1 comparisons.

$O(n)$.

Worst Case:

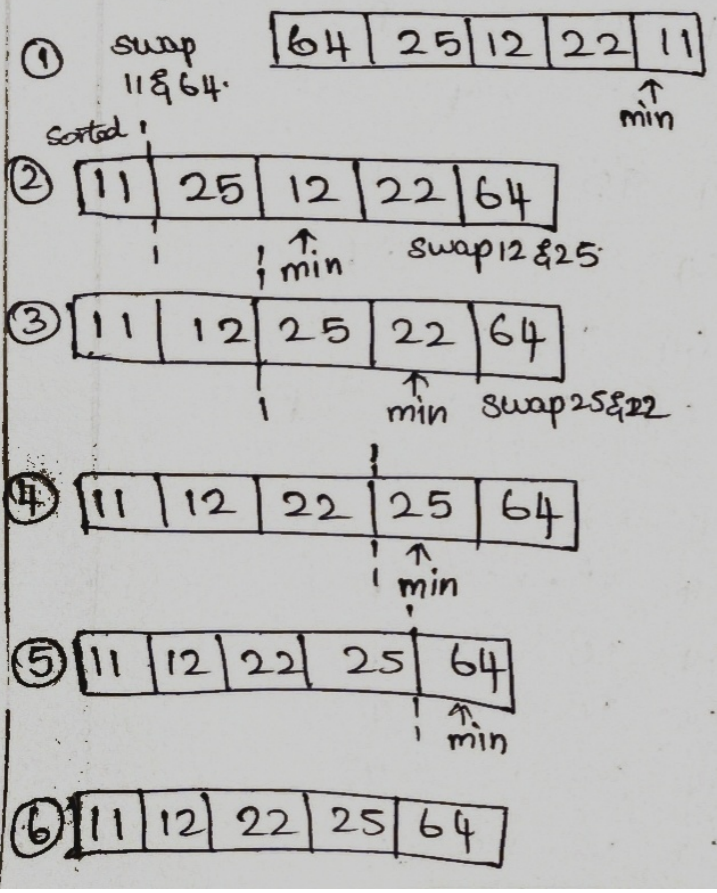
↳ This occurs when array is reverse order. Each pass, Bubble sort makes n-1 comparisons, many swaps,

$\sum_{i=1}^{n-1} i = (n-1)n/2 \Rightarrow O(n^2)$.

Average Case:

↳ Avg. bubble sort will require a quadratic number of Comparisons & swaps. $O(n^2)$.

18. Sort the array 64, 25, 12, 22, 11 using selection sort. what is time complexity of selection sort in best, worst & average cases?



Time Complexity:

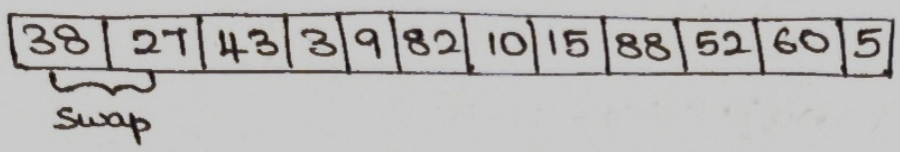
Best case: $O(n^2)$

Worst case: $O(n^2)$

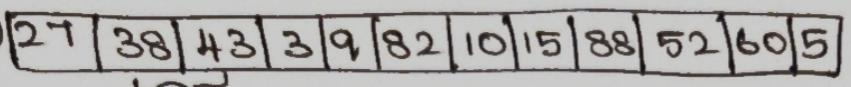
Average case: $O(n^2)$

19. Sort the following elements using Insertion sort using Brute force approach Stategy [38, 27, 43, 3, 9, 82, 10, 15, 88, 52, 60, 5] & analyse complexity of algorithm.

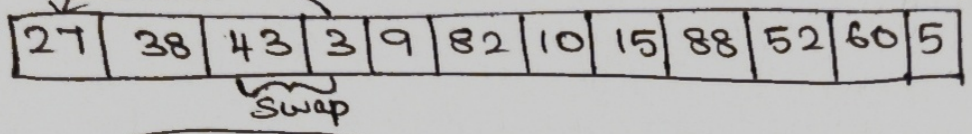
=>



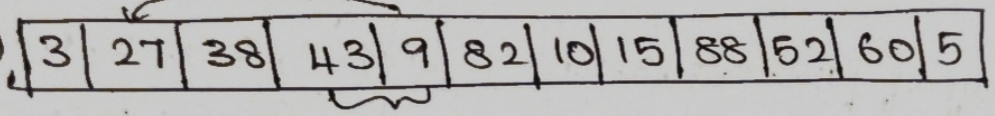
(1)



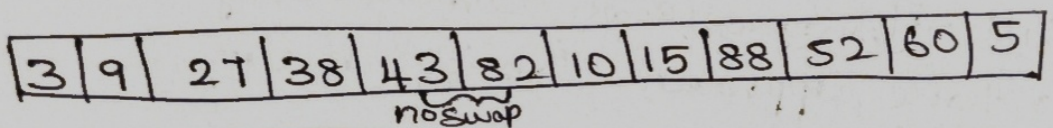
(2)



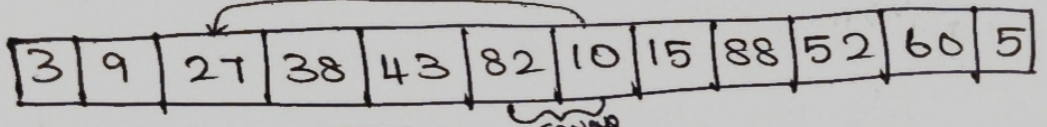
(3)



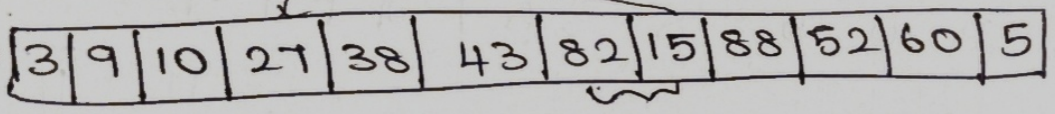
(4)



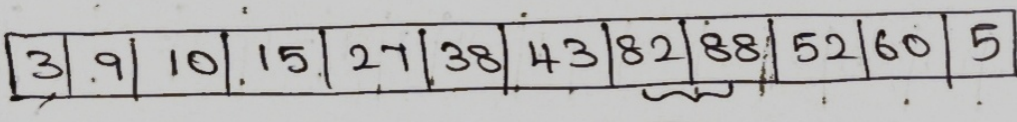
(5)



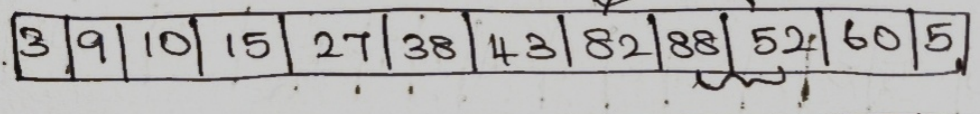
(6)



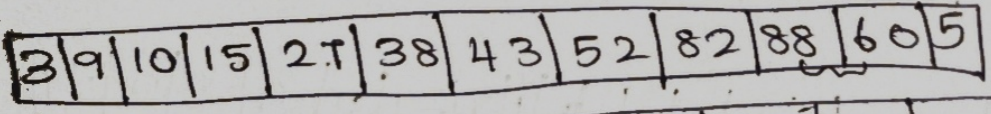
(7)



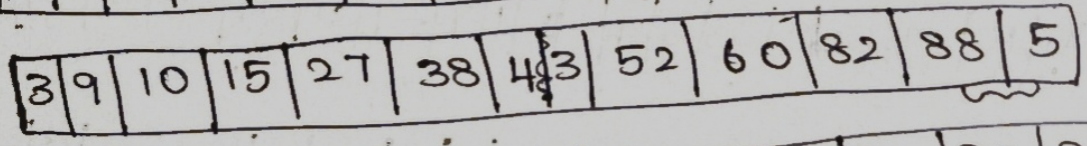
(8)



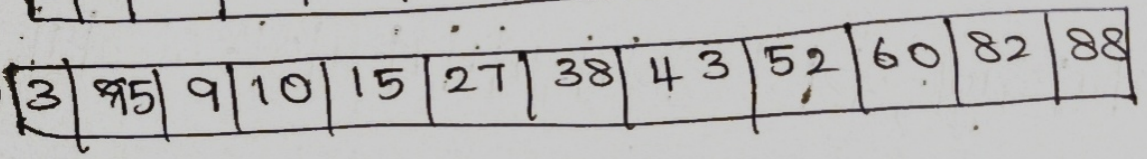
(9)



(10)



(11)



Sorted array: 3, 5, 9, 10, 15, 27, 38, 43, 52, 60, 82, 88.

Time Complexity:

```
Insertionsort (A)
for j = 2 to A.length - n
key = a[j] - 1
i = j - 1 - 1
while i > 0 and aaa[i] > key - n x n
a[i+1] = a[i] - 1
i = i - 1 - 1
a[i+1] = key . - 1
```

$O(n^2)$ //

20. Given an array of [4, -2, 5, 3, 10, -5, 2, 8, -3, 6, 7, -4, 1, 9, -1, 0, -6, -8, 11, -9] integers, sort elements using insertion sort Brute force approach statergy analyze Complexity of algorithm.

Given,

[4 -2 5 3 10 -5 2 8 -3 6 7 -4 1 9 -1 0 -6 -8 11 -9]

① | -2 | 4 | 5 | 3 | 10 | -5 | 2 | 8 | -3 | 6 | 7 | -4 | 1 | 9 | -1 | 0 | -6 | -8 | 11 | -9 |

② | -2 | 4 | 5 | 3 | 10 | -5 | 2 | 8 | -3 | 6 | 7 | -4 | 1 | 9 | -1 | 0 | -6 | -8 | 11 | -9 |

③ | -2 | 3 | 4 | 5 | 10 | -5 | 2 | 8 | -3 | 6 | 7 | -4 | 1 | 9 | -1 | 0 | -6 | -8 | 11 | -9 |

④ | -2 | 3 | 4 | 5 | 10 | -5 | 2 | 8 | -3 | 6 | 7 | -4 | 1 | 9 | -1 | 0 | -6 | -8 | 11 | -9 |

⑤ | -5 | -2 | 3 | 4 | 5 | 10 | 2 | 8 | -3 | 6 | 7 | -4 | 1 | 9 | -1 | 0 | -6 | -8 | 11 | -9 |

⑥ -5 -2 2 3 4 5 10 8 -3 6 7 -4 1 9 -1 0 6 8 11 -9

⑦ -5 -2 2 3 4 5 8 10 -3 6 7 -4 1 9 -1 0 6 8 11 -9

⑧ -5 -3 -2 2 3 4 5 8 10 6 7 -4 1 9 -1 0 6 8 11 -9

⑨ -5 -3 -2 2 3 4 5 6 8 10 7 -4 1 9 -1 0 -6 8 11 -9

⑩ -5 -3 -2 2 3 4 5 6 7 8 10 -4 1 9 -1 0 6 -8 11 -9

⑪ -5 -4 -3 -2 2 3 4 5 6 7 8 10 1 9 -1 0 -6 -8 11 -9

⑫ -5 -4 -3 -2 1 2 3 4 5 6 7 8 10 9 -1 0 -6 -8 11 -9

⑬ -5 -4 -3 -2 1 2 3 4 5 6 7 8 9 10 -1 0 -6 -8 11 -9

⑭ -5 -4 -3 -2 -1 1 2 3 4 5 6 7 8 9 10 0 -6 -8 11 -9

⑮ -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10 -6 -8 11 -9

⑯ -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10 -8 11 -9

⑰ -8 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 -8 9 10 11 -9

⑱ -8 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10 11 -9

⑲ -9 -8 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10 11 -9

Sorted array:

[-9, -8, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]

Complexity:

Best Case $\Rightarrow O(n)$ -> when sorted array. $(n-1)$ comparisons

Average Case $\Rightarrow O(n^2)$

Worst Case $\Rightarrow O(n^2)$. -> $\frac{n(n-1)}{2}$ comparisons.