CSA0666 - Design and Analysis of Algorithm for Divide and Conquer Techniques

S.MD. Waseem Rayan 192373004

Analytical Assignment - 2

1. If $t_1(n) \in O(g_1(n))$ and $t_2(n) \in O(g_2(n))$, prove that $t(n) + t_2(n) \in O(\max\{g_1(n), g_2(n)\})$

$f_1(n) \leq c_1 g_1(n)$ for all $n \geq n_0$

$f_2(n) \leq c_2 g_2(n)$ for all $n \geq n_0$

Adding

$$f_1(n) + f_2(n) \leq c_1 g_1(n) + c_2 g_2(n)$$

Since

$$\max\{g_1(n), g_2(n)\} \geq g_1(n)$$

$$\max\{g_1(n), g_2(n)\} \geq g_2(n)$$

$$f_1(n) + f_2(n) \leq c_1 \max\{g_1(n), g_2(n)\} + c_2 \max\{g_1(n), g_2(n)\}$$

$$\leq (c_1 + c_2) \max\{g_1(n), g_2(n)\}$$

Let $c = c_1 + c_2$

$f_1(n) + f_2(n) \leq c \max\{g_1(n), g_2(n)\}$ for all $n \geq n_0$

$\therefore f_1(n) + f_2(n) = O(\max\{g_1(n), g_2(n)\})$

2) Find the time complexity of the below recurrence equation:

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

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$ By Masters Theorem

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

$K=0$

$1>0$

$\log_b^a > K$

Case i)

$\Theta(n \cdot \log_b^a)$

$\Theta(n \cdot 1)$

$\Theta(n)$

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

Backward Substitution

$T(n) = 2T(n-1) \rightarrow (1)$ Initial $T(0)=0$

$n=n-1$

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

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

sub (2) in (1)

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

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

$n=n-2$

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

$T(n-2) = 2T(n-3) \rightarrow (4)$

sub (4) in (3)

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

$T(n) = 2^3 T(n-3) \rightarrow (5)$

$n=n-3$

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

$T(n-3) = 2T(n-4) \rightarrow (6)$

sub (6) in (5) $\Rightarrow T(n) = 2^3 [2T(n-4)]$

$= 2^4 T(n-4) \rightarrow (7)$

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

$n-k=0 \Rightarrow n=k$

if $T(0)=1$

$T(n) = 2^k \cdot T(0)$

$T(n) = 2^k \cdot 1$

$T(n) = 2^k$

$n=k$

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

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

To Prove that $f(n) = n^2+3n+5$ is $O(n^2)$

We need to find Constants c and no such that $f(n) \le c \cdot n^2$ for all $n \ge n_0$

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

for $n \ge 1$, $n^2 \ge n$ ... so on

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

$f(n) = n^2+3n+5 \le 9n^2$ for $n \ge 1$

So, for $c=9$ and $n_0=1$

$f(n) \le c \cdot n^2$ for all $n \ge n_0$

that proves $f(n)$ is $O(n^2)$

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

To Prove that $g(n) = n^3+2n^2+4n$ is $\Omega(n^3)$

we need to find constant c and no such that

$g(n) \geq c \cdot n^3$ for all $n \geq n_0$

$g(n) = n^3 + 2n^2 + 4n$

for $n \geq 1$,

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

since $2n^2$ and $4n$ ~~are~~ both ~~less~~ than n^3 when $n \geq 1$

so, for $c=1$ and $n_0 = 1$

$g(n) \geq n^3 \cdot c$ for all $n \geq n_0$

that proves $g(n)$ is $\Omega(n^3)$

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

1. $h(n) = 4n^2 + 3n$ is $O(n^2)$

For $n \geq 1$, $h(n) \leq 4n^2 + 3n^2$

(since $3n$ is less than n^2 when $n \geq 1$)

for this simplifies to $h(n) \leq 7n^2$

for $n \geq 1$

therefore, $h(n)$ is $O(n^2)$

2. $h(n) = 4n^2 + 3n$ is $\Omega(n^2)$

for $n \geq 1$, $h(n) \geq 4n^2$

(since $3n$ is positive)

therefore $h(n)$ is $\Omega(n^2)$

since $h(n)$ is both $O(n^2)$ and $\Omega(n^2)$, it is $\Theta(n^2)$

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

$n=1$, $f(1) = 1^3 - 2(1)^2 + 1$
$= 1 - 2 + 1$
$= 0$

$g(1) = (-n)^2$
$= (-1)^2$
$= 1$

$n=2$
$f(2) = 2^3 - \cancel{2}(2)^2 + \cancel{2}$
$= 8 - 8 + 2$
$= 2$

$g(2) = (-2)^2$
$= 4$

$n=3$
$f(3) = 3^3 - 2(3)^2 + 3$
$= 27 - 9 + 3$
$= 21$

$g(3) = (-3)^2$
$= 9$

$n=4$
$f(4) = 4^3 - 2(4)^2 + 4$
$= 64 - 32 + 4$
$= 32 + 4$
$= 36$

$g(4) = (-4)^2$
$= 16$

$n=5$
$f(5) = 5^3 - 2(5)^2 + 5$
$= 15 - 50 + 5$
$= 35 + 5$
$= 40$

$g(5) = (-5)^2$
$= 25$

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

So it is best case according to asymptotic Notation

$f(n) = \Omega(g(n))$

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

1. upper bound (O notation):

we need to find c_1 and n_0 such that

$h(n) \le c_1 \cdot n\log n$ for all $n \ge n_0$

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

$\le n\log n + n\log n$ (since $\log n$ is increasing)

$= 2n\log n$

Now, Let $c_1 = 2$, then $h(n) \le 2n\log n$ for all $n \ge 1$

so, $h(n)$ is $O(n\log n)$

2. <u>Lower bound</u> (Ω Notation):

We need to find c_2 and n_0 such that

$h(n) \ge c_2 \cdot n\log n$ for all $n \ge n_0$

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

$\ge \frac{1}{2} \cdot n\log n$ (for $n \ge 2$)

Now, Let $c_2 = \frac{1}{2}$ then $h(n) \ge \frac{1}{2} \cdot n\log n$

for all $n \ge 2$, so $h(n)$ is $\Omega(n\log n)$

3. <u>Combining Bounds</u>

Since $h(n)$ is both $O(n\log n)$ and $\Omega(n\log n)$

it is also $\Theta(n\log n)$

thus $h(n) = n\log n + n$ is $\Theta(n\log n)$

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

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

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

$a = 4$ $\quad \log_b a = \log_2 4 = 2$

$b = 2$

$K = 2$

$2 = 2$

$\log_b^a = K$

Case iii) $p > -1$ $\Theta(n^K \log_n^{p+1})$

$\Theta(n^2 \cdot \log_n^{1+1})$

$\Theta(n^2 \cdot \log_n^2)$

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

the order of growth for the solution is $n^2 \cdot \log(n)$

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 the maximum and minimum product that can be obtained by multiplying two integers from the array.

Given

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

Maximum product

2 largest no's : 11, 10

2 Smallest (-ve no's) : (-9, -8)

Product :

$11 \times 10 = 110$

$-9 \times -8 = 72$

$\therefore$ max product = 110

Minimum product

$11 \times -9 = -99$

$10 \times -9 = -90$

$\therefore$ Min product $= -99$

12) Demonstrate Binary search Method to search Key=23 from the array arr[] = [2, 5, 8, 12, 16, 23, 38, 56, 72, 91]

arr[] = [2, 5, 8, 12, 16, 23, 38, 56, 72, 91]

Key = 23

0	1	2	3	4	5	6	7	8	9
2	5	8	12	16	23	38	56	72	91

$M = \frac{l+h}{2} = \frac{0+9}{2} = 4.5 \simeq 5$

0	1	2	3	4	5	6	7	8	9
2	5	8	12	16	$\boxed{23}$	38	56	72	91

$\therefore$ arr[mid] = 23

arr[mid] = key

23 = 23

$\therefore$ key is found

13. Apply Merge sort and order the list of 8 elements data d = (45, 67, -12, 5, 22, 30, 50, 20). Set up a recurrence relation for the number of key Comparisons made by merge sort.

	0	1	2	3	4	5	6	7
d =	45,	67,	-12,	5,	22,	30,	50,	20

$M = \frac{0+7}{2} = 4$

0	1	2	3	4	|	5	6	7
45	67	-12	5	22	|	30	50	20

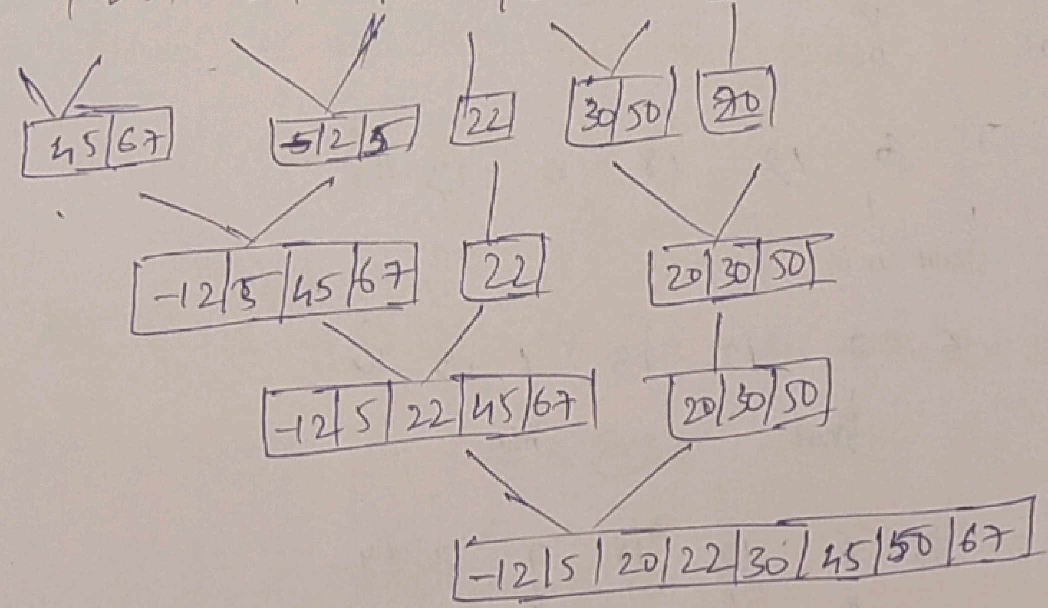
$M = \frac{0+4}{2} = 2$ $\qquad\qquad$ $M=6$

0	1	2		3	4		5	6		7
45	67	-12	|	5	22	|	30	50	|	70

$M = \frac{0+2}{2} = 1$

0	1		2		3		4		5		6		7
45	67	|	-12	|	5	|	22	|	30	|	50	|	20

45 | 67 | -12 | 5 | 22 | 30 | 50 | 20



Recurrence Relation:

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

$a = 2,\ k = 1$

$b = 2,\ p = 1$

$\log_b a = \log_2 2 = 1$

$\Rightarrow \boxed{\log_b a = k}$

$\therefore \Theta(n^k \log_n^{p+1})$

$\Theta(n^1 \log^2 n) \Rightarrow \Theta(n \log n)$

14) Find the no of times to perform swapping for selection sort. Also estimate the time complexity for the order of notation. set S(12, 7, 5, -2, 18, 6, 13, 4)

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

1) 12 5 -2 18 6 13 4

start min

2) -2 . 7 5 12 18 6 13 14

start min

3) -2 5 7 12 18 6 13 14

start min

4) -2 5 6 12 18 7 13 14

start min

5) -2 5 6 7 18 12 13 14

start min

6) -2 5 6 7 12 18 13 14

start min

7) -2 5 6 7 12 13 18 14

start min

8)

-2	5	6	7	12	13	14	18

sorted

11

Time Complexity

Best – $O(n^2)$

Avg – $O(n^2)$ Space Complexity = $O(1)$

Worst – $O(n^2)$

Total no. of swaps = 6

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

Given

0	1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18	20

$M = \frac{l+h}{2} = \frac{0+9}{2} = 4.5 \simeq 5$ or 4

0	1	2	3	4	5	6	7	8	9
2	4	6	8	$\boxed{10}$	12	14	16	18	20
				mid					

target = 10

a[mid] = Target Index = 4

∴ 10 = 10

∴ Target found

16) Sort the following elements using Merge sort divide-and-Conquer strategy [38, 27, 43, 3, 9, 82, 10, 15, 88, 52, 60, 5] and analyse complexity of the algorithm.

Given

0	1	2	3	4	5	6	7	8	9	10	11
38	27	43	3	9	82	10	15	88	52	60	5

$M = \frac{l+h}{2} = \frac{0+11}{2} = 5.5 \simeq 6$

0 1 2 3 4 5 6
38 27 43 3 9 82 10 | 15 88 52 60 5

$M = \frac{0+6}{2} = 3$ $m = \frac{7+11}{2} = 9$

0 1 2 3 4 5 6 7 8 9 10 11
38 27 43 3 | 9 82 10 | 15 88 52 | 60 5

$M = \frac{0+3}{2} = 2$ $m = \frac{4+6}{2} = 5$ $m = \frac{7+9}{2} = 8$ $m = \frac{21}{2} = 10$

0 1 2 3 4 5 6 7 8 9 10 11
38 27 43 | 3 | 9 | 82 | 10 | 15 88 | 52 | 60 | 5

$m = \frac{0+2}{2} = 1$

0 1 2 3 4 5 6 7 8 9 10 11
38 27 | 43 | 3 | 9 | 82 | 10 | 15 | 88 | 52 | 60 | 5

m=0

0 1 2 3 4 5 6 7 8 9 10 11
38 | 27 | 43 | 3 | 9 | 82 | 10 | 15 | 88 | 52 | 60 | 5

27|38   43   3   9|82   10   15|88   52   5|60

27|38|43   3   9|10|82   15|52|88   5|60

3|27|38|43   9|10|82   5|15|52|60|88

3|9|27|38|43|82   5|15|52|60|88

3|9|10|27|38|43|82|5|15|52|60|88

3|5|9|10|15|27|38|43|52|60|82|88

sorted

Time Complexity

Best - $O(n^2)$

Average - $O(n^2)$

Worst - $O(n^2)$

17) Sort the array 64, 34, 25, 12, 22, 11, 90 using Bubble-sort. What is time complexity of Bubble sort in best, worst and Average Cases.

Given

64 34 25 12 22 11 90

It-1

i j

34 64 25 12 22 11 90

i j

34 25 64 12 22 11 90

i j

34 25 12 64 22 11 90

i j

34 25 12 22 64 11 90

i j

34 25 12 22 11 64 90

i j

It-2 34 25 12 22 11 64 90

i j

25 34 12 22 11 64 90

i j

25 12 34 22 11 64 90

i j

25	12	22	34	11	64	90
			i	j		
25	12	22	11	34	64	90
					i	j

It-2

34	25	12	22	11	64	90
i	j					
25	34	12	22	11	64	90
	i	j				
25	12	34	22	11	64	90
		i	j			
25	12	22	34	11	64	90
			i	j		
25	12	22	11	34	64	90
					i	j

It-3

25	12	22	11	34	64	90
i	j					
12	25	22	11	34	64	90
	i	j				
12	22	25	11	34	64	90
		i	j			
12	22	11	25	34	64	90
			i	j		
12	22	11	25	34	64	90
				i	j	
12	22	11	25	34	64	90
					i	j

It – 4

12 22 11 25 34 64 90
i j

12 22 11 25 34 64 90
i j

12 11 22 25 34 64 90
i j

12 11 22 25 34 64 90
i j

12 11 22 25 34 64 90
i j

12 11 22 25 34 64 90
i j

It – 5

12 11 22 25 34 64 90
i j

11 12 22 25 34 64 90
i j

11 12 22 25 34 64 90
i j j

11 12 22 25 34 64 90
i j

$$\underbrace{11 \quad 12 \quad 22 \quad 25 \quad 34 \quad 64 \quad 90}_{\text{sorted}}$$
i j

Time Complexity

Best – $O(n)$

Avg = $O(n^2)$

Worst – $O(n^2)$

18) Sort the array 64, 25, 12, 22, 11 using selection sort. What is time complexity of selection sort in the best, worst and Average Case.

Given

0	1	2	3	4
64	25	12	22	11
start				min

11	25	12	22	64
sorted	start	min		

11	12	25	22	64
sorted		start	min	

11	12	22	25	64

Time Complexity

Best Case : $O(n^2)$

Avg Case : $O(n^2)$

Worst Case : $O(n^2)$

19) Sort the following elements using insertion sort using Brute force approach strategy [38, 27, 43, 3, 9, 82, 10, 15, 88, 52, 60, 5] and analyze complexity of algorithm.

Given

[38 27 43 3 9 82 10 15 28 52 60 5]
i j

1) 27 38 43 3 9 82 10 15 28 52 60 5

2) 27 38 3 43 9 82 10 15 88 52 60 5

3) 27 38 3 43 9 82 10 15 88 52 60 5

4) 3 27 38 43 9 82 10 15 88 52 60 5

5) 3 9 27 38 43 82 10 15 88 52 60 5

6) 3 9 10 27 38 43 82 15 88 52 60 5

7) 3 9 10 15 27 38 43 82 88 52 60 5

8) 3 9 10 15 27 38 43 82 88 52 60 5

9) 3 9 10 15 27 38 43 52 82 60 88 5

10) 3 9 10 15 27 38 43 52 60 82 88 5

11) 3 9 10 15 27 38 43 52 60 82 88 5

12) 3 9 10 15 27 38 43 52 60 82 88 5

13) $\underbrace{3 \quad 5 \quad 9 \quad 10 \quad 15 \quad 27 \quad 38 \quad 43 \quad 52 \quad 60 \quad 82 \quad 88}_{\text{sorted}}$

Time Complexity

Best case - O(n)) - This occurs when the array is already sorted. The inner loop will run only once

Avg case - O(n²)) - This list is randomly ordered

Worst case - O(n²)) If the list is in reverse

Space Complexity

O(1) - Insertion sort.

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 the following elements using insertion sort using brute force approach strategy analyze complexity of algorithm.

Given array

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\underbrace{-9\ -8\ -6\ -5\ -4\ -3\ -2\ -1\ 0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9}_{\text{sorted}}\ 10\ 11$$

Time Complexity

Best Case ($O(n)$) - this occurs the array is already sorted
The inner loop will run only once for each element.
Average Case ($O(n^2)$) - The list is randomly ordered
Worst Case ($O(n^2)$) - If the list is in reverse order.