R. Santhosh
192211471

CSA0669 - Design and Analysis of Algorithm for polynomial Problems.

DATE
10/06/2024
Assignment - 2

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

For any four arbitrary real numbers, a_1, b_1, a_2, b_2 such that $a_1 \le b_1$ and $a_2 \le b_2$

we have $a_1 + a_2 \le 2 \max\{b_1, b_2\}$

since $t_1(n) \in O(g_1(n))$, then there exists some constant c_1 and non-negative integer n_1 such that

$$t_1(n) \le c_1 g_1(n) \text{ for all } n \ge n_1$$

since $t_2(n) \in O(g_2(n))$, then there exists some constant c_2 and non-negative integer n_2 such that

$$t_2(n) \le c_2 g_2(n) \text{ for all } n \ge n_2$$

Let $c_3 = \max\{c_1, c_2\}$ and $n_0 = \max\{n_1, n_2\}$

$$\begin{aligned} t_1(n) + t_2(n) &\le c_1 g_1(n) + c_2 g_2(n) \\ &\le c_3 g_1(n) + c_3 g_2(n) \\ &= c_3 \{g_1(n) + g_2(n)\} \\ &\le 2 c_3 \max\{g_1(n), g_2(n)\} \end{aligned}$$

Hence $t_1(n) + t_2(n) \le O(\max\{g_1(n), g_2(n)\})$, with constant $c = 2c_3 = 2\max\{c_1, c_2\}$ and n_0 required by the O definition being $n_0 = \max\{n_1, n_2\}$ respectively.

2. Find the time complexity of the follow recurrence equation:

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

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

$a = 2$

$b = 2$

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

$k = 0$

$\log_b^a = k$

case 1)

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

$\Theta(n \cdot 1)$

$\Theta(n)$

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

Backward sub.

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

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

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

• $n = n-3$

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

$$T(n-3) = 2T(n-4) \rightarrow (5)$$

sub (6) in (5)

$$T(n) = 2^3[2T(n-4)]$$

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

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

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

if $T(0) = 1$

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

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

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

$$n = k$$

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

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

To prove that $f(n) = n^2 + 3n + 5$ if $O(n^2)$ we need to find constants c and n_0 such that.

$f(n) \leq c \cdot n^2$ for all $n \geq n_0$

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

for $n \geq 1$, $n^2 \geq n$ so on

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

$f(n) = n^2 + 2n + 5 \leq 9n^2$ for $n \geq 1$

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

$f(n) \leq cn^2$ for all $n \geq n_0$

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

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 constants c and n_0 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 $n_0 = 1$

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

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

Big that notation

1. $h(n) = 4n$

for $n \geq 1$,

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

1. Upper Bound (O notation):

we need to find c_1 and n_0 such that

$h(n) \leq c_1 n\log n$ for all $n \geq n_0$

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

$\leq n\log n + n\log n$ (since $\log n$ is n (clearly))

$= 2n\log n$

now let $c_1 = 2$, then $h(n) \leq 2n\log n$ for all $n \geq 1$

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

2. Lower bound (Ω notation):

we need to find c_2 and n_0 such that

$h(n) \geq c_2 n\log n$ for all $n \geq n_0$

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

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

now, let $c_2 = \frac{1}{2}$, then $h(n) \geq \frac{1}{2} n\log n$

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

3. Combining Bounds.

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 in $\Theta(n\log n)$

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

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

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

$a = 4$

$b = 2$

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

$k = 2$

$2 = 2$

$\log_b^a = k$

Case II) $p > -1 \quad \Theta(n^k \log_n^{p+1})$

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

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

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

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

n=1

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

$= 8 - 8 + 1$

$= 1$

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

$= 1$

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

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

$= 125 - 50 + 5$

$= 75 + 5$

$= 80$

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

So it is best case according to asymptotic notation.

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

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 $\Theta(n^2)$

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

$n = 1$

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

$= 1 - 2 + 1$

$= 0$

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

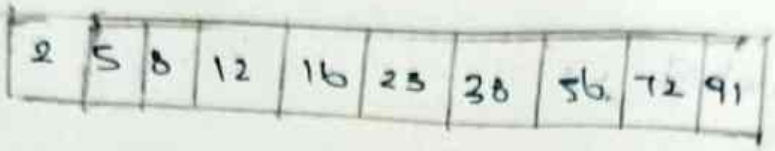
$= (-1)^2$

$= 1$

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

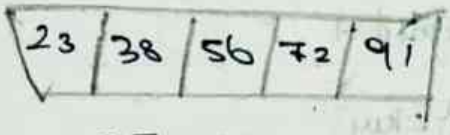
```
def find_m(arr):
    if len(arr)<2:
        raise ValueError("Array atleast constain 2 element")
    arr.sort()
    max_product = max(arr[-1]*arr[-2], arr[0]*arr[1])
    max_product = min(arr[-1]*arr[0], arr[0]*arr[1])
    return max_product, min_product

arr = [4, -2, 5, 3, 10, -5, 2, 8, -3, 6, 7, -4, 1, 9, -1, 0, -6, 8, 11, -9]
max, min_product = find_m(arr)
print(find_m)
```

Berrians take the binary search method to search key=23 from the array a[] = {2, 5, 8, 12, 16, 23, 38, 56, 72, 91}



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

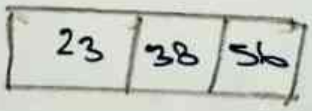
a[mid] == key
a[4] == 23
16 != 23
16 < 23
low = mid + 1



low = 5 high = 9
mid = $\frac{5+9}{2} = \frac{14}{2} = 7$

a[mid] == key
a[7] == 23
56 != 23 . 56 > 23

low = 5 high = 7
mid = $\frac{5+7}{2} = 6$



a[6] == 23
38 != 23
38 > 23

low = 5 high = 5
mid = $\frac{5+5}{2} = 5$

high = ~~mid~~ key
23 == 23

Return the position of the key (i.e) 5

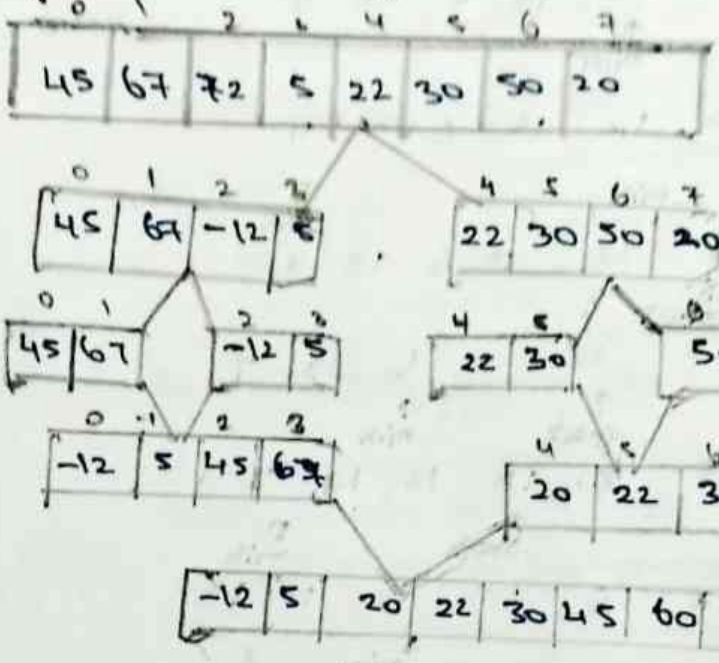
pseudocode:

```
binary_search (a, n, key)
    low = 0
    high = n-1
while (low <= high);
    mid = (high + low)/2
    if a[mid] == key
        return mid
    a[mid] > key
        high = mid - 1
    a[mid] < key
        low = mid + 1
    return -1 (If not found)
```

Time complexity

O(nlogn)

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



∴ this sorted list is:

-12, 5, 20, 22, 30, 45, 60, 67

Time complexity: $O(n \log n)$

Recurrence relation: $T(n) = 2T(n/2) + (n-1)$

2 9 10 15 27 28 43 52 5 60 82 88

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

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

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

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

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

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

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

Sort the array 64, 34, 25, 12, 22, 11, 90 using Bubble sort and its the time complexity of selection sort in best, average worst case).

64 34 25 12 22 11 90

34 64 25 12 22 11 90

34 25 64 12 22 11 90 phase-I

34 25 12 64 22 11 90

34 25 12 22 64 11 90

34 25 12 22 11 64 90

34 25 12 22 11 64 90

---

34 25 12 22 11 64 90

25 34 12 22 11 64 90

25 12 34 22 11 64 90 phase-II

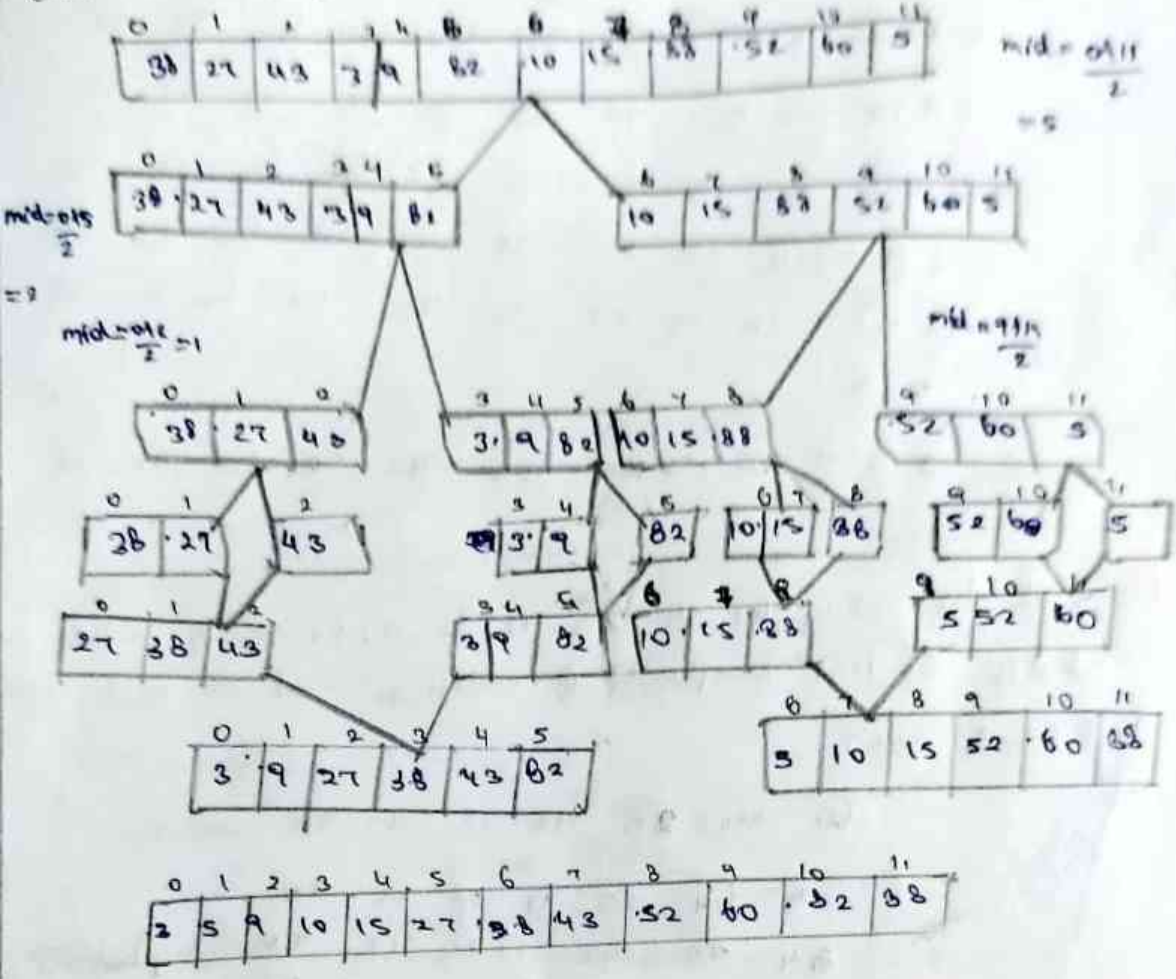
25 12 22 34 11 64 90

25 12 22 11 34 64 90

25 12 22 11 34 64 90

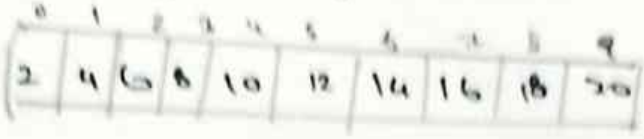
solve the elements using merge sort divide conquer strategy

(38, 27, 43, 3, 9, 82, 10, 15, 88, 52, 60, 5) analyze the time complexity.



∴ The sorted list is: 3, 5, 9, 10, 15, 27, 38, 43, 52, 60, 82, 88

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



low = 0
high = 9
mid = (0+9)/2
= 4

a[mid] == key

a[4] == 10

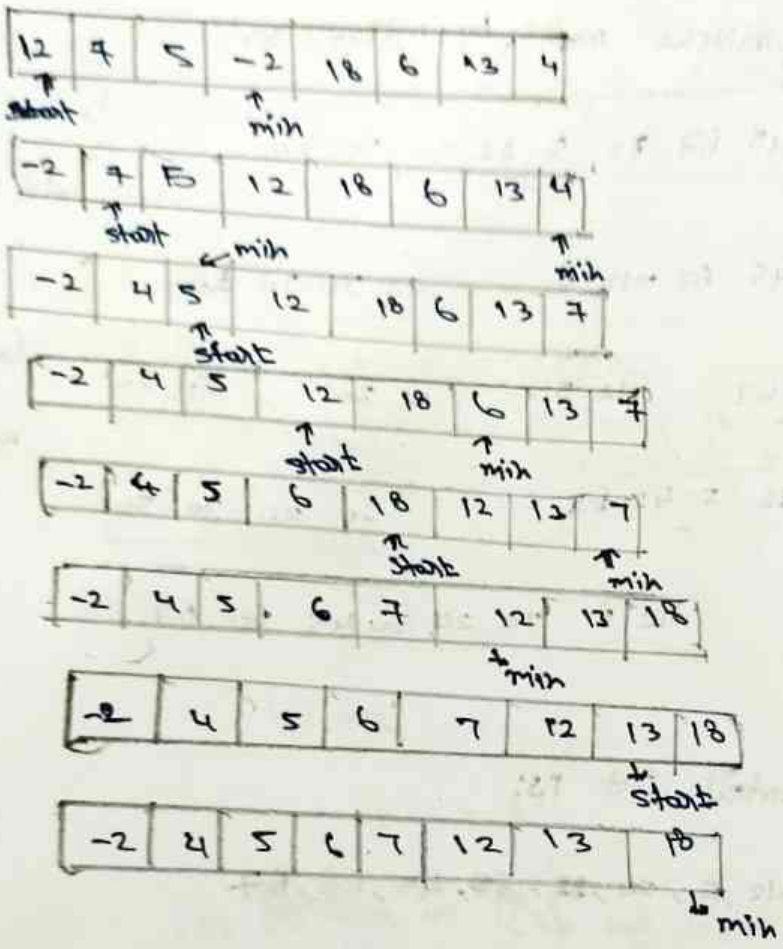
10 == 10

Return the position of the key (i.e) 4

Pseudocode:

```
binary-search (array, size of array, key)
    low = 0
    high = size - 1
    while (low <= high)
    mid = (high + low)/2
        if a[mid] == key
            return mid
        a[mid] < key
            high = mid - 1
        a[mid] > key
            low = mid + 1
    return -1   [if not found]
```

Find the no. of times to perform swapping for selection sort
Also estimate the time complexity S2[12, 7, 5, -2, 18, 6, 13, 4]



sorted list: -2, 4, 5, 6, 7, 12, 13, 18

usually, the number of swaps required will be n-1.
But for this question there are only 4 swaps.

Time complexity: $O(n^2)$ It is n^2 in all the three cases.

25 12 22 11 34 64 90

12 25 22 11 34 64 90 — phase - III

12 22 25 11 34 64 90

12 22 11 25 34 64 90

12 22 11 25 34 64 90

---

12 22 11 25 34 64 90

12 22 11 25 34 64 90 — phase - II

12 11 22 25 34 64 90

11 12 22 25 34 64 90

---

11 12 22 25 34 64 90 — phase - I

11 12 22 25 34 64 90

The sorted list: 11, 12, 22, 25, 34, 64, 90

Time complexity: $O(n^2)$

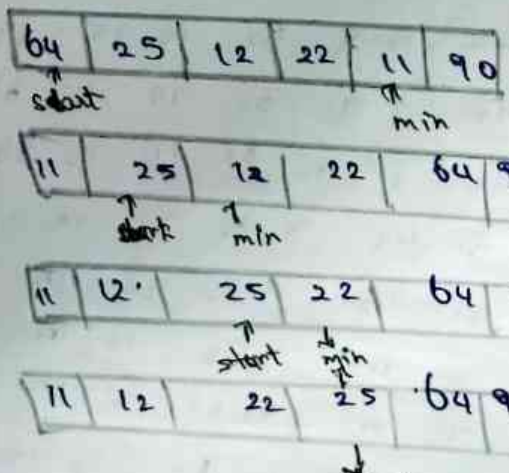
selection sort: sort case - $O(n^2)$

worst case - $O(n^2)$

Average case: $O(n^2)$

---

sort the array 64, 25, 12, 22, 11 using sort what is the time complexity.



given an array sort using insertion sort

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

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

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

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

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

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

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

time complexity $\Rightarrow O(n^2)$

space complexity $\Rightarrow O(1)$

Pseudocode:

```
insertion sort (a, size of a):
    for i in range (1, size):
        key = a[i]
        j = i - 1
        while j >= 0 and a[j] > key :
            a[j+1] = a[j]
            j -= 1
        a[j+1] = key
    return a
```

Time complexity:

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

$$= n^2 - n + 4$$

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

time complexity is $O(n^2)$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

The sorted list is: 3, 5, 9, 10, 15, 27, 38, 43, 52, 60, 82, 88

11	12	22	25	64	90

11	12	22	25	64	90

The sorted list : 11, 12, 22, 25, 64, 90

The complexity :

The time complexity of the selection sort Algorithm is always $O(n^2)$. because, for each element in the array, the algorithm iterate through the remaining elements to the minimum.

The outer loop n+1 times and the inner loops n-1, n-2 till, time

Best case: $O(n^2)$

Worst case: $O(n^2)$

Average case: $O(n^2)$

---

solve the following using insertion sort using fore approach (38, 27, 43, 3, 9, 82, 10, 15, 88, 52, 60, 5).

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

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

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

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

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

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

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

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