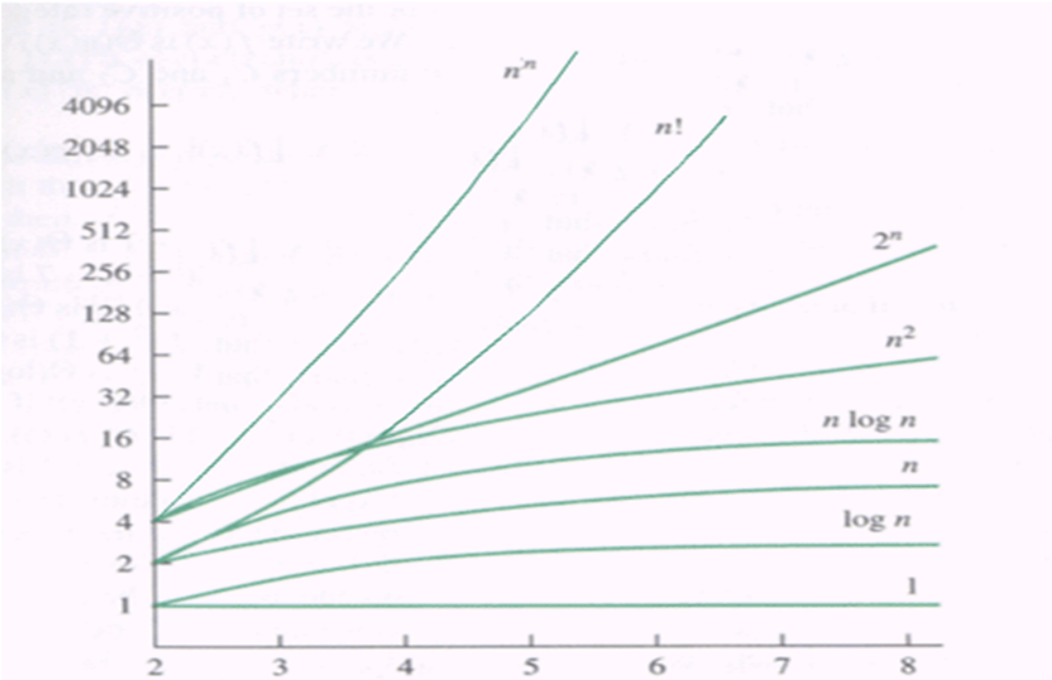

1.

$T(n) = n^2$ since, $n^2 > n\log(n)$

$T(n) = 2^n$ since $2^n > n^8 > n^3 > n$

2.

j = 0 ← constant time

for i = 0 to n ←n times

while j ≤ 5 ←n x constant time(because the condition of the while loop is nt based on the input size)

j =j +1 ←n x constant time(because the condition of the while loop is nt based on the input size)

Total steps = 1+ n + 7n + 6n = 14n + 1

Therefor, T(n) = n

3.

	INSERTION-SORT(A)	Cost	Times
1	**for** j = 2 **to** A.length	c_1	**n**
2	key = A[j]	c_2	**n-1**
3	// Insert A[j] into the sorted // sequence A[1..j-1]	0	**n-1**
4	i = j – 1	c_4	**n-1**
5	**While** i > 0 **and** A[i] > key	c_5	$\sum^n_{j=2} t_j$
6	A[i+1] = A[i]	c_6	$\sum^n_{j=2} (t_j - 1)$
7	i = i-1	c_7	$\sum^n_{j=2} (t_j - 1)$
8	A[i+1] = key	c_8	**n-1**

Let t_i be the number of times the while loop was executed for that value of i

From the above diagram we can conclude that the running time can be given as

$$T(n) = c_1 n + c_2 (n-1) + c_4 (n-1) + c_5 \sum_{j=2}^{n} t_j + c_6 \sum_{j=2}^{n} (t_j - 1) + c_7 \sum_{j=2}^{n} (t_j - 1) + c_8(n-1)$$

By simplifying it for the best case and the worse case.

Best Case (Array is in sorted order)

- $T(n) \leftarrow an+b$

The time complexity therefore is, $\Theta(n)$

•Worst Case (Array is in reverse sorted order)

- $T(n) \leftarrow cn^2 + dn + e$

The time complexity therefore is, $\Theta(n^2)$

4.

1	2	3	4
3	4	1	2

1) A.length = 4

j = 2

2) i = 1

3) A[j] > A[i] ← 4>3 T

4) i = 2

3) A[j] > A[i] ← 4>4 F

5) Key = 4

6) k = 0 and j-i-1 = 0

8) A[i] = A[2] = 4

1) j = 3

2) i = 1

3) A[3] > A[1] ← 1>3 F

5) key = A[3] = 1

6) k = 0 to j-i-1 =1

7) A[j-k] = A[j-k-1] = A[3] = A[2],

A[3] = 4

6) when k = 1 the loop doesn't execute

8) A[1]=1

1	2	3	4
1	3	4	2

1) j = 4

2) i = 1

3) A[4] > A[1] ← 2>1 T

4) i = 2

3) A[4] > A[2] ← 2>3F

5) key = A[4] = 2

6) A[4-0] = A[4-0-1]

7) A[4-1] = A[4-1-1]

8) A[2] = 2

1	2	3	4
1	2	3	4