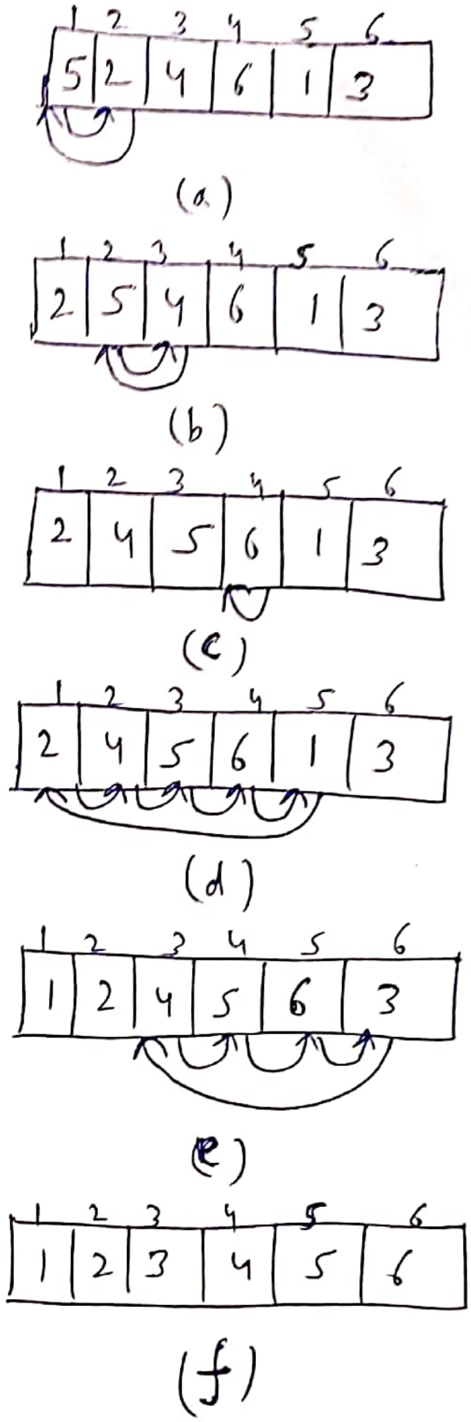
Correctness of INSERTION-SORT

INSERTION-SORT (A)

```
1) for j = 2 to A.length
2)        key = A[j]
3)        i = j-1
4)        while i>0 and A[i] > key
5)              A[i+1] = A[i]
6)              i = i-1
7)        A[i+1] = key
```



Step(I) : Identify "Loop Invariant"

At the beginning of each iteration of the for loop [lines 1 to 7], the subarray $A[1 \ldots j-1]$ are the elements originally in $A[1 \ldots j-1]$, but in sorted order.

Step(II) : Initialization

Prior to the first iteration of the for loop, the subarray $A[1 \ldots j-1]$, contains only one element i.e. $A[1 \ldots 1] = A[1]$, which is trivially in sorted order.

Step(III): Maintenance

The body of the for loop works by moving $A[j-1]$, $A[j-2]$, $A[j-3]$, ... and so on, by one position to the right until it finds the proper position for $A[j]$; the point at which it inserts the value of $A[j]$.
The subarray $A[1 \ldots j]$ then consists of the elements originally in $A[1 \ldots j]$, but now in sorted order.
Incrementing j for the next iteration; preserves the loop invariant.

Step(IV): Termination

The loop terminates when $j > A.length = n$.

i.e. $\boxed{j = n+1}$

So, the subarray $A[1 \ldots j-1] = A[1 \ldots n+1-1]$
$= A[1 \ldots n]$ consists of the elements originally in $A[1 \ldots n]$, but now in sorted order. Hence, the entire array $A[1 \ldots n]$ is now sorted.

So, INSERTION-SORT(A) is correct.

Analysis of INSERTION-SORT (Time Complexity of INSERTION-SORT)

INSERTION-SORT(A)

	cost	No. of times
for j = 2 to A.length	c_1	n
key = A[j]	c_2	$n-1$
i = j − 1	c_3	$n-1$
while i > 0 and A[i] > key	c_4	$\sum_{j=2}^{n} t_j$
A[i+1] = A[i]	c_5	$\sum_{j=2}^{n} (t_j - 1)$
i = i − 1	c_6	$\sum_{j=2}^{n} (t_j - 1)$
A[i+1] = key	c_7	$n-1$

The running time $T(n)$ of INSERTION-SORT(A) can be expressed as:-

$$T(n) = c_1 n + c_2(n-1) + c_3(n-1) + c_4 \sum_{j=2}^{n} t_j + c_5 \sum_{j=2}^{n} (t_j - 1) + c_6 \sum_{j=2}^{n} (t_j - 1) + c_7(n-1)$$

$$= c_1 n + c_2 n - c_2 + c_3 n - c_3 + c_4 \sum_{j=2}^{n} t_j + c_5 \sum_{j=2}^{n} (t_j - 1) + c_6 \sum_{j=2}^{n} (t_j - 1) + c_7 n - c_7$$

$$= (c_1 + c_2 + c_3 + c_7) n + (-c_2 - c_3 - c_7) + c_4 \sum_{j=2}^{n} t_j + c_5 \sum_{j=2}^{n} (t_j - 1) + c_6 \sum_{j=2}^{n} (t_j - 1)$$

Best-case Analysis (when array is already sorted)

In this case, "while" loop statement $A[i] > key$ is always FALSE, because $A[i] \leq key$ is satisfied.

Thus, $\boxed{t_j = 1}$

So, the running time can be expressed as :-

$$T(n) = (c_1+c_2+c_3+c_7)n + (-c_2-c_3-c_7) + c_4(n-1)$$
$$= (c_1+c_2+c_3+c_7)n + (-c_2-c_3-c_7) + c_4 n - c_4$$
$$= (c_1+c_2+c_3+c_7+c_4)n + (-c_2-c_3-c_7-c_4)$$

which is of the form $\underline{an+b}$ (linear function of n).

Worst-case Analysis (when array is in descending order)

In this case, $A[i] > key$ is always TRUE.

Thus, $\boxed{t_j = j}$

So, $$\sum_{j=2}^{n} j = \frac{n(n+1)}{2} - 1$$

and $$\sum_{j=2}^{n} (j-1) = \frac{n(n-1)}{2}$$

So, $$T(n) = (c_1+c_2+c_3+c_7)n + (-c_2-c_3-c_7) + c_4\left[\frac{n(n+1)}{2} - 1\right] + c_5\left[\frac{n(n-1)}{2}\right] + c_6\left[\frac{n(n-1)}{2}\right]$$

$$T(n) = (c_1+c_2+c_3+c_7)n + (-c_2-c_3-c_7) + c_4\left(\frac{n^2+n}{2} - 1\right) + c_5\left(\frac{n^2-n}{2}\right) + c_6\left(\frac{n^2-n}{2}\right)$$

$$= \left(\frac{c_4}{2} + \frac{c_5}{2} + \frac{c_6}{2}\right)n^2 + \left(c_1+c_2+c_3+c_7+\frac{c_4}{2} - \frac{c_5}{2} - \frac{c_6}{2}\right)n + (-c_2-c_3-c_7-c_4)$$

which is of the form $\underline{\underline{an^2+bn+c}}$ (quadratic function of n)