Ejercicio 1

Dado el siguiente algoritmo que realiza la ordenación de un array de enteros *a*, donde *n* representa la dimensión del array.

```
void InsertionSort(int a[],int n)
{
  int j, p;
  int temp;
  for(p=1;p<n;p++)
  {
     temp = a[p];
     for(j=p;j>0 && a[j-1]>temp;j--)
     {
          a[j] = a[j-1];
          a[j-1] = temp;
      }
   }
}
```

SE PIDE:

1. Describir qué es el tamaño del problema. Justificar la respuesta.
2. Describir qué caso es el mejor y peor para ese algoritmo y calcular el T(n) de cada uno de dichos casos.
3. Al ser un algoritmo iterativo que proviene de un algoritmo recursivo final. Indicar el código en lenguaje C de dicho algoritmo recursivo final.
4. Calcular el T(n) de este nuevo algoritmo para el caso mejor y peor.

Puntuación: 3 puntos

SOLUCIÓN

1. Describir qué es el tamaño del problema. Justificar la respuesta.
   *El tamaño del problema viene determinado por el parámetro **n** que representa la dimensión del array, dado que la condición del primer bucle depende del valor de dicho parámetro. Esto conduce que a medida que se haga este parámetro mayor la complejidad temporal será mayor también.*

2. Describir qué caso es el mejor y peor para ese algoritmo y calcular el T(n) de cada uno de dichos casos.
   El caso mejor es cuando el vector este ordenado ascendentemente o todos los elementos del vector son iguales pues entonces la condición a[j-1] >temp es siempre falsa y por tanto no se ejecuta el cuerpo del segundo bucle. Por tanto

$$T(n) = \sum_{p=1}^{n-1} k = k(n-1) \in \theta(n)$$

   El caso peor es cuando el array viene ordenado descendentemente donde siempre la condición a[j-1]> temp es verdadera y por tanto se ejecuta el cuerpo del bucle interno.

$$T(n) = \sum_{p=1}^{n-1} \sum_{j=0}^{p} k = \sum_{p=1}^{n-1} k(p+1) = k(n-1) * \frac{n}{2} + k(n-1) \in \theta(n^2)$$

3. Al ser un algoritmo iterativo que proviene de un algoritmo recursivo final. Indicar el código en lenguaje C de dicho algoritmo recursivo final.

El algoritmo podría ser reescrito con el bucle más externo mediante un while para facilitar la visualización de las características que podría tener el algoritmo recursivo final del que procede este algoritmo iterativo.

```
void InsertionSort(int a[],int n)
{
  int j, p=1;
  int temp;
  while(p<n)
  {
     temp = a[p];
     for(j=p;j>0 && a[j-1]>temp;j--)
     {
          a[j] = a[j-1];
          a[j-1] = temp;
      }
   p++;
   }
}
```

Esto ayuda a determinar el caso base del algoritmo recursivo final como la negación de (p<n), la solución al caso base como vacía y el siguiente elemento en la recursión p++, quedando entonces la función recursiva final como:

```
void InsertSortRec(int a[], int p, int n)
{

   int j, temp;

   if(p<n)
   {
     temp = a[p];
     for(j=p;j>0 && a[j-1]>temp;j--)
     {
          a[j] = a[j-1];
          a[j-1] = temp;
      }
     InsertSortRec(a,p+1,n);
   }
}
```

*Para que sea equivalente al algoritmo iterativo del que procede la llamada a este algoritmo recursivo debería ser **InserSortRec(a,0,n),** siendo n el tamaño del array, como en el algoritmo iterativo.*

5. Calcular el T(n) de este nuevo algoritmo para el caso mejor y peor.

Hay que tener ahora en cuenta que para el algoritmo recursivo el tamaño es n-p, pues ambos parámetros determinan la condición de finalización, de acuerdo a la condición que aparece en la sentencia if. Para la primera llamada al ser p=0 entonces tamaño es n-0=n. Y el siguiente problema al incrementar p en 1, el tamaño será ya n-1.

El caso mejor es cuando el vector este ordenado ascendentemente o todos los elementos del vector son iguales pues entonces la condición a[j-1] >temp es siempre falsa y por tanto no se ejecuta el bucle que existe en este algoritmo recursivo. Entonces

$$T(n) = T(n-1) + k$$

Que es un caso particular de este tipo de ecuaciones:

$$T(n) = aT(n-b) + p(n)$$

Siendo en general $p(n) = \Theta(n^d)$, *la solución es:*

$$T(n) \in \begin{cases} \Theta\left(a^{n/b}\right), & si \quad a > 1 \\ \Theta(n^{d+1}), & si \quad a = 1 \\ \Theta(n^{d}), & si \quad a < 1 \end{cases}$$

Por tanto la solución es T(n) $\in \theta$ (n)

El caso peor es cuando el array viene ordenado descendentemente donde siempre la condición a[j-1]> temp es verdadera y por tanto se ejecuta el cuerpo del bucle del algoritmo recursivo hasta n veces cuando p=n. Entonces

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

Por tanto la solución es T(n) $\in \theta$ (n^2)