INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

Práctica 3

Divide y vencerás

Unidad académica: Análisis de Algoritmos
Profesor a cargo: Dra. Sandra Díaz Santiago
Grupo: 3CV1
Realizada por:

- Medina Juárez Jesús Booz
- Ríos Altamirano Alam Yael

Sesión de laboratorio: 7 de marzo del 2018

Ejercicio 1

Dado un arreglo ordenado ascendentemente, A de n enteros distintos, se requiere hallar el índice i, tal que $A[i] = i$. Diseña un algoritmo cuya complejidad sea $O(\log n)$. Algoritmo implementado en Python 2.7

```
def busquedabin(arr, bajo, alto):
    mid = (bajo + alto)//2
    if mid is arr[mid]:
        return mid
    if mid > arr[mid]:
        return busquedabin(arr, (mid + 1), alto)
    else:
        return busquedabin(arr, bajo, (mid -1))
    return -1 # Regresa -1 si no encuentra un valor
```

```
arr = [-3, -1, 2, 100]
n = len(arr)
print(str(busquedabin(arr, 0, n-1)))
```

```
$python main.py
2
```

```
arr = [0, 2, 4, 100]
n = len(arr)
print(str(busquedabin(arr, 0, n-1)))
```

```
$python main.py
0
```

```
arr = [0, 3, 4, 5, 6, 7, 13, 54, 201]
n = len(arr)
print(str(busquedabin(arr, 0, n-1)))
```

```
$python main.py
0
```

```
arr = [-1, 0, 1, 2, 3, 4, 5, 6, 8]
n = len(arr)
print(str(busquedabin(arr, 0, n-1)))
```

```
$python main.py
8
```

Lo que hace el algoritmo es dividir el arreglo en subarreglos de n/2 y en el peor de los casos 3 comparaciones antes de la llamada recursiva.

Ejercicio 2

Diseña un algoritmo que dado un arreglo con los coeficientes de un polinomio $p(x) = a_o + a_1x + a_2x^2 + \cdots + a_nx^n$ y un valor a, devuelva la evaluación del polinomio en a, es decir, el valor de $p(a)$. El algoritmo debe tener una complejidad menor al algoritmo trivial $O(n^2)$

Algoritmo implementado en Python 2.7

```
def evaluarPolinomio(A, x):
    p = A[-1]
    i = len(A) - 2
    while i >= 0:
        p = p * x + A[i]
        i -= 1
    return p
```

```
 RESTART: E:\ESCUELA\SextoSemest
actica\evaluarPolinomio.py
a: [1, 2] x_0: 1 p(x_0): 3
>>>
 RESTART: E:\ESCUELA\SextoSemest
actica\evaluarPolinomio.py
a: [2, 5, 2] x_0: 2 p(x_0): 20
>>>
 RESTART: E:\ESCUELA\SextoSemest
actica\evaluarPolinomio.py
a: [2, 5, 2] x 0: 4 p(x 0): 54
```

Podemos observar que el tiempo polinomial del algoritmo es:

$$T(n) = C_1 + C_2 + (n-1)(C_3 + C_4) + C_5$$
$$T(n) = an + b$$

Y al definir una asíntota superior para la expresión

$$T(n) = O(n)$$
$$an + b \leq cn$$

Definiendo $c = a + b$ y $n \geq 1$

$$an + b \leq (a + b)n$$

Es evidente que es cierto y por lo tanto podemos afirmar

$$T(n) = O(n)$$

Y se sencillo notar que realiza menos operaciones que el algoritmo por fuerza bruta $O(n^2)$

Ejercicio 3

Dado un arreglo A de n enteros distintos, se dice que dos índices forman una inversión si $i < j$ y $A[i] > A[j]$

- Algoritmo por fuerza bruta implementado en Python 2.7

```
def inversiones(A):
    indices = []
    for i in range(0,len(A)-1):
        for j in range(i+1, len(A)):
            if(A[i] > A[j]):
                indices.append((i,j))
    return indices
```

```
 RESTART: E:\ESCUELA\SextoSemestre\Algoritmos\Practicas\P
actica\inversionesFB.py
A: [3, 5, 1, 4, 9] inversiones: [(0, 2), (1, 2), (1, 3)]
>>>
 RESTART: E:\ESCUELA\SextoSemestre\Algoritmos\Practicas\P
actica\inversionesFB.py
A: [2, 4, 1, 3, 5] inversiones: [(0, 2), (1, 2), (1, 3)]
>>>
 RESTART: E:\ESCUELA\SextoSemestre\Algoritmos\Practicas\P
actica\inversionesFB.py
A: [2, 4, 1] inversiones: [(0, 2), (1, 2)]
```

Podemos notar que el tiempo polinomial del algoritmo es:

$$T(n) = C_1 + \left(\frac{n^2 - n}{2}\right)(C_3 + C_4) + C_5 n + C_6$$

$$T(n) = an^2 + bn + c$$

$$an^2 + bn + c \leq dn^2$$

$$\rightarrow d = a + b + c$$

$$\rightarrow n_0 \geq 1$$

$$\therefore an^2 + bn + c \leq (a + b + c)n^2$$

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

- Algoritmo con adaptación de mezclado para mejorar la complejidad del algoritmo

```
class InversionesMerge
{

  /* Ordena la matriz de Entrada y
devuelve la cantidad de inversiones */
  static int mergeSort(int arr[], int
array_size)
  {
     int temp[] = new int[array_size];
     return _mergeSort(arr, temp, 0,
array_size - 1);
  }

  /* Auxiliar que ordena la matriz y regresa
la cantidad */
  static int _mergeSort(int arr[], int temp[],
int left, int right)
  {
    int mid, inv_count = 0;
    if (right > left)
    {
     /* divide en 2 el arreglo y llama al
contador de inversiones */
      mid = (right + left)/2;

      /* el contador de inversiones sera la
suma de las inversiones en ambas partes y
el numero de inversiones en la union */
      inv_count  = _mergeSort(arr, temp,
left, mid);
      inv_count += _mergeSort(arr, temp,
mid+1, right);

      /*juntar ambas partes*/
      inv_count += merge(arr, temp, left,
mid+1, right);
    }
    return inv_count;
  }

  /* junta ambas partes y devuelve el
conteo de inversiones*/
  static int merge(int arr[], int temp[], int
left, int mid, int right)
  {
    int i, j, k;
    int inv_count = 0;

    i = left; /* i indice del la parte
izquierda*/
    j = mid;  /* j indice del la parte
derecha*/
    k = left; /* k indice resultante del la
union*/
    while ((i <= mid - 1) && (j <= right))
    {
     if (arr[i] <= arr[j])
     {
      temp[k++] = arr[i++];
     }
     else
     {
      temp[k++] = arr[j++];
      inv_count = inv_count + (mid - i);
     }
    }

    /* copia los elementos de la izquierda*/
    while (i <= mid - 1)
     temp[k++] = arr[i++];

    /* copia los elementos de la derecha*/
    while (j <= right)
     temp[k++] = arr[j++];

    /*hace el arreglo original*/
    for (i=left; i <= right; i++)
     arr[i] = temp[i];

    return inv_count;
  }

  public static void main(String[] args)
  {
    int arr[] = new int[]{1, 20, 6, 4, 5};
    System.out.println("La cantidad de
inversiones son: " + mergeSort(arr, 5));

  }
}
```

```
    int arr[] = new int[]{6, 12, 6, 4, 3};
    System.out.println("La cantidad de inversiones son: " + mergeSort(arr, 5));
```

```
Run:
La cantidad de inversiones son: 8
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
 int arr[] = new int[]{0, 1, 2, 3, 4};
 System.out.println("La cantidad de inversiones son: " + mergeSort(arr, 5));
```

```
La cantidad de inversiones son: 0
BUILD SUCCESSFUL (total time: 0 seconds)
```