Análisis de Algoritmos 2022-1
Tarea 4

Alumnos:
Hernández Sánchez Oscar José
Altamirano Niño Luis Enrique

17 de junio de 2022

Ejercicios

- Da un algoritmo que lo resuelva en tiempo $O(n \log n)$.

  Respuesta:

Algoritmo 1 Variante del algoritmo merge que también se encarga de contar el número de parejas ordenadas tal que uno de los elementos de la pareja pertenece a la primera mitad, y el otro es un elemento de la segunda mitad.

```
def merge(A,B):
    i=j=0
    C = []
    cuenta = 0
    while i < len(A) and j < len(B):
        if A[i] <= B[j]:
            C.append(A[i])
            i += 1
            cuenta += (len(B)-j)
        else:
            C.append(B[j])
            j += 1
    C += A[i:]
    C += B[j:]
    return cuenta,C
```

Algoritmo 2 Algoritmo que se encarga de contar las parejas ordenadas en un arreglo, además regresa el arreglo ordenado ya que se requiere para el merge.

```
def cuenta(A):
    n = len(A)
    if n < 2: return 0,A
    #Partimos el arreglo en dos.
    L = A[:n//2]
    R = A[n//2:]
    #Contamos las parejas de ambas mitades por separado.
    cL, L = cuenta(L)
    cR, R = cuenta(R)
    #Contamos las parejas mezclando ambas mitades.
    cA, A = merge(L,R)
    return cL+cR+cA, A
```

Algoritmo 3 Función principal que solo se encarga de regresar la cuenta del número de parejas ordenadas.

```
def cuentaPares(A):
    return cuenta(A)[0]
```

- Demuestra que tu algoritmo es correcto.

  Demostración:

  Por inducción en el tamaño del arreglo.

  Caso base: Si el tamaño del arreglo es 0 o 1, entonces el algoritmo regresa 0, que en efecto es el número de parejas ordenadas en un arreglo de cualquier de esos tamaños.

  Hipótesis de inducción: Supongamos que el algoritmo regresa el número de parejas ordenadas para un arreglo de tamaño a lo más k.

  Paso inductivo: Debemos mostrar que el algoritmo regresa el número de parejas ordenadas para un arreglo de tamaño $k+1$. Entonces el algoritmo parte el arreglo en dos, y se manda a llamar recursivamente con cada mitad del arreglo y como el tamaño de nuestro arreglo es $k+1$, entonces el tamaño de la mitad izquierda será $\lfloor \frac{k+1}{2} \rfloor$ y el de la derecha será $\lceil \frac{k+1}{2} \rceil$, y es claro que $\lfloor \frac{k+1}{2} \rfloor, \lceil \frac{k+1}{2} \rceil \leq k$, por lo que por la hipótesis de inducción ambas llamadas recursivas regresarán el resultado correcto. Después el algoritmo se encarga de contar las parejas ordenadas mezclando ambas mitades y las suma con la cuenta de cada mitad por separado. Por lo que al mezclar ambas mitades, volvemos a obtener el arreglo de tamaño $k+1$, y al hacer las sumas, obtenemos el número de parejas ordenadas en el arreglo.

  Por lo tanto el algoritmo es correcto. ■

- Demuestra la complejidad del algoritmo.

  Respuesta:

  Dado que la modificación del algoritmo merge solo agrega unas lineas que toman tiempo constante, entonces nuestra modificación sigue tomando tiempo $O(n)$, de igual forma partir el arreglo en dos toma tiempo $O(n)$, y además construyendo el árbol de recursión en el que la raíz es el arreglo de tamaño n, y cada uno de los hijos izquierdo y derecho son cada una de las mitades del arreglo padre, entonces en

general procesar cada uno de los niveles del árbol tomará tiempo $O(n)$ ya que:

$$\begin{aligned} O(n) &= O(n) \text{ (Nivel 1)} \\ O(n/2) + O(n/2) &= O(n) \text{ (Nivel 2)} \\ O(n/4) + O(n/4) + O(n/4) + O(n/4) &= O(n) \text{ (Nivel 3)} \\ &\vdots \end{aligned}$$

y dado que el algoritmo tiene como caso base un arreglo de tamaño 1, entonces la altura del árbol será $\log_2 n$, por lo que el costo total del algoritmo será la suma de todos los costos de procesamiento de los vértices del árbol de recursión, sin embargo, como procesar cada uno de los niveles del árbol tomará tiempo $O(n)$, entonces el costo total del algoritmo será $\underbrace{(O(n) + \cdots + O(n))}_{\log_2 n \text{ veces}} = O(n) \cdot \log_2 n = O(n \log n)$.