Algoritmos y Estructuras de Datos II

Guía 4

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Ejercicios obligatorios de la práctica

Integrante	LU	Correo electrónico
Bruno, Patricio Damián	62/19	pdbruno@gmail.com

Reservado para la cátedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

1. Ordenamiento

sortSets(**in** A: `arreglo(ConjuntoDeNaturales))` $\rightarrow$ res: arreglo(ConjuntoDeNaturales))

1: $K \leftarrow obtenerMaximoCardinal(A)$ ▷ $O(NK)$
2: $cardinales : arreglo\_dimensionable\ de\ lista(puntero(ConjuntoDeNaturales))$
3: $cardinales \leftarrow crearArreglo(K)$ ▷ $O(K)$
4: **for** $i \leftarrow 1$ to K **do** ▷ $O(K)$
5: $\quad cardinales[i] \leftarrow Vacia()$ ▷ $O(1)$
6: **end for**
7: **for** $i \leftarrow 1$ to $tam(A)$ **do** ▷ $O(N)$
8: $\quad cardinal \leftarrow obtenerCardinal(A[i])$ ▷ $O(K)$
9: $\quad AgregarAtras(cardinales[cardinal], \&A[i])$ ▷ $O(1)$
10: **end for**
11: $res : arreglo\_dimensionable\ de\ ConjuntoDeNaturales$
12: $res \leftarrow crearArreglo(tam(A))$ ▷ $O(N)$
13: $j \leftarrow 1$ ▷ $O(1)$
14: **for** $i \leftarrow 1$ to K **do** ▷ $O(K)$
15: $\quad it \leftarrow CrearIt(cardinales[i])$
16: $\quad$ **while** $HaySiguiente?(it)$ **do** ▷ $O(N)$ en total
17: $\quad\quad res[j] \leftarrow *Siguiente(it)$ ▷ $O(asdasd)$
18: $\quad\quad j \leftarrow j + 1$ ▷ $O(1)$
19: $\quad\quad Avanzar(it)$ ▷ $O(1)$
20: $\quad$ **end while**
21: **end for**
22: **return** res

Justificación: Las primeras dos líneas son asignaciones de enteros, $\Theta(1)$. La guarda del if es una comparación de enteros, la instrucción siguiente es una asignación de enteros (recordar que acceder a una posición de un arreglo toma tiempo constante) y el bloque del else implica evaluar el máximo entre dos enteros y asignarlo a un variable, todo esto tambíen es $\Theta(1)$. La inicialización de B toma tiempo lineal en función de su tamaño, M. Si A tiene un elemento igual a n-1, estaremos en el peor caso y M tomará su mayor valor posible: n-1. El en el mejor caso, todos los elementos son 0 o mayores o iguales a n, y M valdrá 0. En el peor cado reemplazo $\Theta(M)$ por $\Theta(n)$ y en el mejor, por $\Theta(1)$

$T_{mejor}(n) = \Theta(1) + \Theta(1) + \sum_{i=0}^{n} \Theta(1) + \Theta(1) + \Theta(1)$
$= \Theta(1) + n\,\Theta(1)$
$= \Theta(n)$

$T_{peor}(n) = \Theta(1) + \Theta(1) + \sum_{i=0}^{n} \Theta(1) + \Theta(n) + \sum_{i=0}^{n} \sum_{j=i}^{n} \Theta(1)$
$= \Theta(n) + \Theta(1) \sum_{i=0}^{n} (n - i)$
$= \Theta(n) + \Theta(1)(\sum_{i=0}^{n} n - \sum_{i=0}^{n} i)$
$= \Theta(n) + \Theta(1)(n^2 - \frac{n(n+1)}{2})$

$n^2 - \frac{n(n+1)}{2} = \frac{n^2}{2} - \frac{n}{2} \in \Theta(n^2)$ (*)
$= \Theta(n) + \Theta(n^2)$
$= \Theta(n^2)$

(*) $\frac{n^2}{2} - \frac{n}{2} \in O(n^2)$ trivialmente, ya que la primer función es siempre menor
$\frac{n^2}{2} - \frac{n}{2} \in \Omega(n^2) \Leftrightarrow \exists k, n_0 \geq 0 / n \geq n_0 \Rightarrow \frac{n^2}{2} - \frac{n}{2} \geq k\, n^2$
$\frac{n^2}{2} - \frac{n}{2} \geq k\, n^2 \Rightarrow \frac{1}{2} - \frac{1}{2n} \geq k$ Tomemos $k = \frac{1}{4}$
$\frac{1}{2} - \frac{1}{2n} \geq \frac{1}{4} \Rightarrow \frac{1}{4} \geq \frac{1}{2n} \Rightarrow 2 \leq n$ Entonces alcanza con tomar $n_0 = 2$

---

obtenerMaximoCardinal(**in** A : `arreglo(ConjuntoDeNaturales)`) $\rightarrow$ res: nat

```
1: res ← −1                                          ▷ O(1)
2: for i ← 1 to tam(A) do                            ▷ O(N)
3:     card ← obtenerCardinal(A[i])                  ▷ O(K)
4:     res ← max(res, card)                          ▷ O(1)
5: end for
6: return res
```

Justificación: Las primeras dos líneas son asignaciones de enteros, $\Theta(1)$. La guarda del if es una comparación de enteros, la instrucción siguiente es una asignación de enteros (recordar que acceder a una posición de un arreglo toma tiempo constante) y el bloque del else implica evaluar el máximo entre dos enteros y asignarlo a un variable, todo esto tambíen es $\Theta(1)$. La inicialización de B toma tiempo lineal en función de su tamaño, M. Si A tiene un elemento igual a n-1, estaremos en el peor caso y M tomará su mayor valor posible: n-1. El en el mejor caso, todos los elementos son 0 o mayores o iguales a n, y M valdrá 0. En el peor cado reemplazo $\Theta(M)$ por $\Theta(n)$ y en el mejor, por $\Theta(1)$

---

obtenerCardinal(**in** C : `ConjuntoDeNaturales`) $\rightarrow$ res: nat

1: $res \leftarrow 0$ ▷ $O(1)$
2: $it \leftarrow CrearIt(C)$ ▷ $O(1)$
3: **while** $HaySiguiente?(it)$ **do** ▷ $O(K)$
4: $res[j] \leftarrow res + 1$ ▷ $O(1)$
5: $Avanzar(it)$ ▷ $O(1)$
6: **end while**
7: **return** res

Justificación: Las primeras dos líneas son asignaciones de enteros, $\Theta(1)$. La guarda del if es una comparación de enteros, la instrucción siguiente es una asignación de enteros (recordar que acceder a una posición de un arreglo toma tiempo constante) y el bloque del else implica evaluar el máximo entre dos enteros y asignarlo a un variable, todo esto tambíen es $\Theta(1)$. La inicialización de B toma tiempo lineal en función de su tamaño, M. Si A tiene un elemento igual a n-1, estaremos en el peor caso y M tomará su mayor valor posible: n-1. El en el mejor caso, todos los elementos son 0 o mayores o iguales a n, y M valdrá 0. En el peor cado reemplazo $\Theta(M)$ por $\Theta(n)$ y en el mejor, por $\Theta(1)$

$T_{mejor}(n) = \Theta(1) + \Theta(1) + \sum_{i=0}^{n} \Theta(1) + \Theta(1) + \Theta(1)$
$= \Theta(1) + n\,\Theta(1)$
$= \Theta(n)$

$T_{peor}(n) = \Theta(1) + \Theta(1) + \sum_{i=0}^{n} \Theta(1) + \Theta(n) + \sum_{i=0}^{n} \sum_{j=i}^{n} \Theta(1)$
$= \Theta(n) + \Theta(1) \sum_{i=0}^{n} (n - i)$
$= \Theta(n) + \Theta(1)(\sum_{i=0}^{n} n - \sum_{i=0}^{n} i)$
$= \Theta(n) + \Theta(1)(n^2 - \frac{n(n+1)}{2})$
$n^2 - \frac{n(n+1)}{2} = \frac{n^2}{2} - \frac{n}{2} \in \Theta(n^2)$ (*)
$= \Theta(n) + \Theta(n^2)$
$= \Theta(n^2)$

(*) $\frac{n^2}{2} - \frac{n}{2} \in O(n^2)$ trivialmente, ya que la primer función es siempre menor
$\frac{n^2}{2} - \frac{n}{2} \in \Omega(n^2) \Leftrightarrow \exists k, n_0 \geq 0 / n \geq n_0 \Rightarrow \frac{n^2}{2} - \frac{n}{2} \geq k\,n^2$
$\frac{n^2}{2} - \frac{n}{2} \geq k\,n^2 \Rightarrow \frac{1}{2} - \frac{1}{2n} \geq k$ Tomemos $k = \frac{1}{4}$
$\frac{1}{2} - \frac{1}{2n} \geq \frac{1}{4} \Rightarrow \frac{1}{4} \geq \frac{1}{2n} \Rightarrow 2 \leq n$ Entonces alcanza con tomar $n_0 = 2$

2. Dividir y Conquistar

2.1.

sonDisjuntos(**in** C : `arreglo(ConjuntoDeNaturales)`) $\rightarrow$ res: bool

1: $res \leftarrow sonDisjuntosRec(C, 1, tam(C))$
2: **return** $res.sonDisjuntos$

sonDisjuntosRec(**in** C: `arreglo(ConjuntoDeNaturales)`, **in** $inicio$: `nat`, **in** fin: `nat`) $\rightarrow$ res: $< sonDisuntos : bool, union : arreglo/listadenaturales >$

```
1:  if fin − inicio = 1 then                                          ▷ O(1)
2:      hacerlacosa                                                   ▷ O(1)
3:  return < true, cosa >
4:  else
5:      med ← (inicio + fin)/2                                        ▷ O(1)
6:      izq ← sonDisjuntosRec(C, inicio, med)                         ▷ O(T(n/2))
7:      der ← sonDisjuntosRec(C, med, fin)                            ▷ O(T(n/2))
8:      disjuntos ← true                                              ▷ O(1)
9:      union : arreglo_dimensionable de ConjuntoDeNaturales
10:     union ← crearArreglo(tam(der.union) + tam(izq.union))         ▷ O(N ∗ M)
11:     i, j, k ← 1                                                   ▷ O(1)
12:     while k ≤ tam(union) ∧ disjuntos do                           ▷ O(N ∗ M)
13:         if i ≤ tam(izq.union) ∧ j ≤ tam(der.union) ∧ izq.union[i] =
        der.union[j] then
14:             disjuntos ← false                                     ▷ O(1)
15:         end if
16:                                  ▷ el resto es el algoritmo clásico de merge de arreglos
17:         if j > tam(der.union) ∨ (i ≤ tam(izq.union) ∧ izq.union[i] <
        der.union[j]) then
18:             union[k] ← izq.union[i]                               ▷ O(1)
19:             i ← i + 1                                             ▷ O(1)
20:         else
21:             union[k] ← der.union[j]                               ▷ O(1)
22:             j ← j + 1                                             ▷ O(1)
23:         end if
24:         k ← k + 1                                                 ▷ O(1)
25:     end while
26:
27: end if
28: return < disjuntos ∧ izq.disjuntos ∧ der.disjuntos, union >
```

Justificación: Las primeras dos líneas son asignaciones de enteros, $\Theta(1)$. La guarda del if es una comparación de enteros, la instrucción siguiente es una asignación de enteros (recordar que acceder a una posición de un arreglo toma tiempo constante) y el bloque del else implica evaluar el máximo entre dos enteros y asignarlo a un variable, todo esto tambíen es $\Theta(1)$. La inicialización de B toma tiempo lineal en función de su tamaño, M. Si A tiene un elemento igual a n-1, estaremos en el peor caso y M tomará su mayor valor posible: n-1. El en el mejor caso, todos los elementos son 0 o mayores o iguales a n, y M valdrá 0. En el peor cado reemplazo $\Theta(M)$ por $\Theta(n)$ y en el mejor, por $\Theta(1)$

2.2.

sonDisjuntos(**in** C : `arreglo(ConjuntoDeNaturales)`) $\rightarrow$ res: bool

```
1:  elementos : arreglo_dimensionable de bool
2:  M ← obtenerCardinal(C[1])                                  ▷ O(M)
3:  elementos ← crearArreglo(tam(C) * M))                      ▷ O(N * M)
4:  for i ← 1 to tam(elementos) do                             ▷ O(N * M)
5:      elementos[i] ← false                                   ▷ O(1)
6:  end for
7:  res ← true                                                 ▷ O(1)
8:  i ← 1                                                      ▷ O(1)
9:  while i ≤ tam(C) ∧ res do                                  ▷ O(N)
10:     it ← CrearIt(C[i])                                     ▷ O(1)
11:     while HaySiguiente?(it) ∧ res do                       ▷ O(M)
12:         elem ← Siguiente(it)                               ▷ O(1)
13:         if elementos[elem] then                            ▷ O(1)
14:             res ← false                                    ▷ O(1)
15:         else
16:             elementos[elem] ← true                         ▷ O(1)
17:         end if
18:         Avanzar(it)                                        ▷ O(1)
19:     end while
20:     i ← i + 1                                              ▷ O(1)
21: end while
22: return res
```

Justificación: Las primeras dos líneas son asignaciones de enteros, $\Theta(1)$. La guarda del if es una comparación de enteros, la instrucción siguiente es una asignación de enteros (recordar que acceder a una posición de un arreglo toma tiempo constante) y el bloque del else implica evaluar el máximo entre dos enteros y asignarlo a un variable, todo esto tambíen es $\Theta(1)$. La inicialización de B toma tiempo lineal en función de su tamaño, M. Si A tiene un elemento igual a n-1, estaremos en el peor caso y M tomará su mayor valor posible: n-1. El en el mejor caso, todos los elementos son 0 o mayores o iguales a n, y M valdrá 0. En el peor cado reemplazo $\Theta(M)$ por $\Theta(n)$ y en el mejor, por $\Theta(1)$