Tarea 2 - Parte Teórica - Arboles y Grafos

Juliana Sanchez Gomez

Agosto 2024

Contents

1 Algoritmo 1

1.1 Entrada y salida

Entrada: $N \in \mathbb{N}$
Salida: $A = [a_0 \dots a_N) : \forall_t t \in [0 \dots N) \cdot A(t) = (t+1)!$
I_0: $1 \leq i \leq N+1$
I_1: $ac = i!$
I_2: $ans = [a_0 \dots a_{i-1}) \forall_t t \in [0 \dots i-1) \cdot ans[t] = (t+1)!$

No deben separarse con /\

Teorema???

Demostración: ....

1.2 Inicialización

De acuerdo a la primera linea tenemos $i = 1 \land ans = [] \land ac = 1$ para iniciar.
Si tomamos I_0 entonces $1 \leq 1 \leq N+1$. En el peor de los casos N es 0 y $0+1 = 1$, se cumple la invariante.

¿el peor de los casos?

Si tomamos I_1 entonces $1! = 1 \Rightarrow 1 = 1$. Se cumple la invariante.
Si tomamos I_2 entonces $ans = [0 \dots 0) \Rightarrow$ lo que implica que tiene 0 elementos, entonces se cumple la invariante.

¿por qué? ¿cómo?

1.3 Estabilidad

Se asume que antes de alguna iteración $i = j$ que se cumplen las invariantes:
$1 \leq j \leq N+1 \land ac = j! \land ans[0 \dots j-1) \forall_t t \in [0j-1) \cdot ans(t) = (t-1)!$.
Se quiere mostrar que antes de la iteración $i = j+1$ las invariantes se cumplen y por lo tanto estas expresiones se cumplen:
I_0: $1 \leq j+1 \leq N+1$
I_1: $ac = (j+1)!$
I_2: $ans = [a_0 \dots a_{(j+1)-1}) \Rightarrow ans = [a_0 \dots a_j) \forall_t t \in [0 \dots j) \cdot ans(t) = (t+1)!$
Al ejecutar las lineas 4, 5 y 6:

```
ans.append(ac)
ac = ac * (i + 1)
i = i + 1
```

ac

ans[t]

con los valores $i = j$ y $a = j!$; y un arreglo $ans = [a_0 \dots a_{j-1}) \forall_t t \in [0 \dots j-1) \cdot ans(t) = (t+1)!$ se tiene:
$ans = [a_0 \dots a_{j-1}, a_j)$ donde $ans[j-1] = j! \Rightarrow$ parla.

???

$ac = j! * (j+1)$
$i = j+1$

Luego el invariante I_0 es verdadero ya que $1 \leq j+1 \leq N+1$ si se asumió que $1 \leq j \leq N+1$, entonces $j+1$ seguirá siendo mayor a 1. Por el otro lado, como asumimos que con $i = j$ se pudo entrar nuevamente al ciclo, entonces el valor máximo que pudo tomar j es N, de esta manera $j+1$ es menor o igual a $N+1$.
Adicionalmente $ac = j! * (j+1) = (j+1)!$ cumple con la Invariante 1. Por

se debería explicar más formalmente

ultimo al haberle agregado j! al arreglo quedo de j elementos de manera que cumple con la I_2.
Finalmente, es en la iteración en la que termina el ciclo que se tiene que $i = N + 1$, luego por la estabilidad del invariante 1 se satisface que $ans = [a_0 \dots a_N) \forall_t t \in [0 \dots N) \cdot ans(t) = (t+1)!$ y es conservancia cuando se llega a la linea 7 se cumple la salida.

Teorema de correctitud?

¿por qué se sabe que el ciclo termina?

2 Algoritmo 2

2.1 Entrada y salida

???

Entrada: $N \in \mathbb{N} \wedge N \geq 2$
Salida: $A \in [a_0, a- \dots a_n]$ Un arreglo de los factores primos del valor de entrada N.
I_0: $2 \leq i \leq N$
I_1: $1 \leq N^{'} \leq N \longrightarrow$ Siendo $N^{'}$ el N que va variando.
I_2: $ans = [a_0 \dots a_k] \longrightarrow$ Siendo $0 \leq k \leq log_2 N$.
I_3: $\left(\prod_{i=0}^{k} a_i\right) \cdot N^{'} = N$

Teorema???

Demostración: ....

2.2 Inicialización

De acuerdo a las lineas 2 y 3:

```
i = 2
ans[], N' = N
```

Si tomamos I_0: $2 \leq i \leq N \longrightarrow 2 \leq 2 \leq N \longrightarrow$ Se cumple
Si tomamos I_1: $1 \leq N^{'} \leq N \longrightarrow 1 \leq N \leq N \longrightarrow$ Se cumple
Si tomamos I_2: $ans = [a_0 \dots a_k)$ *es* $k = 0$
Si tomamos I_3: $\prod_{i=0}^{k} ans[i] \cdot N^{'} = N \longrightarrow \left(\prod_{i=0}^{0} ans[i]\right) \cdot N = N \rightarrow$ *Por propiedad de la multiplicatoria* $\rightarrow$ $1 \cdot N = N \longrightarrow N = N$

Esto es incorrecto. El número de la iteración no coincide con el valor de i ya que i en muchas iteraciones no cambia.

2.3 Estabilidad

Asumir antes de alguna iteración $i = j$ que se cumplen las invariantes: $2 \leq j \leq N \wedge 1 \leq N^{'} \leq N \wedge ans[a_0 \dots a_k) \wedge \prod_{i=0}^{k} a_i \cdot N^{'} = N$

Se quiere mostrar que antes de la iteración $i = j + 1$ las invariantes se cumplen si $N^{'} \% j \neq 0$ solo varia j por lo que solo haría falta verificar I_0 $2 \leq j + 1 \leq N$, las demás se mantienen. Al ejecutar la linea 5, al no cumplirse el condicional salta a la linea 10:

```
i = i + 1
```

Luego el invariante I_0 es verdadero ya que si asumimos que se puede entrar en

el ciclo entonces el i no ha llegado a ser igual a $N^{'}$, pues habría entrado en el primer condicional y $N^{'}$ hubiera quedado igual a 1. Por tanto:
$2 \leq j+1 \leq N$.
Luego si $N^{'}\%j = 0$, se asume que se cumplen los demás invariantes, j ya no varia (se mantiene el I_0).
I_1: $1 \leq N^{'} \leq N$
I_2: $ans[a_0 \dots a_k)$
I_3: $\left(\prod_{i=0}^{k} a_i\right) \cdot N^{'} = N$
Al ejecutar las lineas 6 y 7:

```
ans.push_back(i)
N = N/i
```

Con los valores $ans[a_0 \dots a_k]$ y $i = j \wedge N = N^{'} \wedge ans[a_0 \dots a_k, i] \wedge N = N^{'}/i$
Luego el invariante I_1 se cumple pues en el peor de los casos el i por el que $N^{'}$ puede ser dividido es el mismo $N^{'}$, por lo tanto $1 \leq N^{'} \leq N$. Así mismo, el I_2 también se cumple, ya que se sigue teniendo un arreglo de m elementos siendo m un entero. $(k+1)$ es un entero.
Por ultimo I_3: $\left(\prod_{i=0}^{k+1} a_i\right) \cdot N^{'}/i = N$ conocemos $a_k = i$ por lo tanto I_3:
$\left(\prod_{i=0}^{k} a_i\right) \cdot i \cdot \frac{N^{'}}{i} = N$
I_3: $\left(\prod_{i=0}^{k} a_i\right) \cdot N^{'} = N$. El invariante I_3 se cumple. Se sabe que el ciclo termina pues al irle sumando de 1 en 1 a i, en algún momento deberá ser igual a N, y como $N\%N = 0$ se dividirá N en N, ya que hará $N^{'} = 1$, incumpliendo la condición del ciclo y llevándolo a detenerse.
Finalmente, en la iteración en la que termina el ciclo se tiene que i es igual al ultimo elemento agregado a ans el cual tuvo el mismo valor de $N^{'}$ antes de ser dividido por i, por lo que $N^{'} = 1$. Luego, por la estabilidad del I_3, se satisface:
$\left(\prod_{i=0}^{k} a_i\right) \cdot N^{'} = N \rightarrow \left(\prod_{i=0}^{k} a_i\right) \cdot 1 = N \rightarrow \prod_{i=0}^{k} a_i = N$
Y en conservancia cuando se llega a la linea 12 se cumple la salida esperada.

Teorema de correctitud?

¿por qué se sabe que el ciclo termina?

3 Algoritmo 3

3.1 Entrada y salida

La forma correcta sería A = [a0, a1, ..., a_(N-1)]

Entrada: $A = [a_0 \dots a_N)$
Salida: $A = [a_0 \dots a_N)$ con los mismos valores de A ordenados ascendentemente, es decir, $\forall_i i \in [0 \dots N-1) \cdot A[i] \leq A[i+1]$
I_0: $0 \leq i \leq N$ siendo N el numero de elementos de A el arreglo de entrada.
I_1: El arreglo A se encuentra ordenado ascendentemente hasta $i-1$ y todos los

A[0 .. i) está ordenado ascendentemente ...

tmp = l[pos]

demás elementos son mayores o iguales a el ultimo valor.
Para demostrar que los invariantes I_0, I_1 se cumplen en ese rango es necesario poder establecer que el ciclo while realiza su trabajo apropiadamente. Para esto se requiere hacer un análisis de invariantes para este ciclo, analizando las lineas 4 - 9.
I_2: $i + 1 < j < N$
I_3: *pos* corresponde siempre a la posicion en el arreglo del valor de *tmp*.
I_4: *tmp* es el menor valor del subarreglo $A[i \dots j)$
Teorema invariantes I_2, I_3 y I_4. Los invariantes se cumplen.

Demostración: ...

3.2 Inicialización $(I_2, I_3 y I_4)$

De acuerdo a la linea 4:

```
j = i + 1
tmp = A[i]
pos = i
```

Si tomamos $I_2 : i + 1 \leq j \leq N \rightarrow i + 1 \leq i + 1 \leq N \rightarrow$ se cumple.
Si tomamos $I_3 : i \leq pos \leq N - 1 \rightarrow i \leq i \leq N - 1 \rightarrow$ se cumple.
Si tomamos $I_4 tmp$ es el menor valor del subarreglo de $A[i \dots j)$, $A[i \dots i + 1) \cdot tmp = A[i]$, es el único elemento, por lo tanto se cumple.

Por lo tanto, ...

3.3 Estabilidad $(I_2, I_3 y I_4)$

Se considera una iteración arbitraria en la que $j = k$ y se asumen que antes de ejecutar esta iteración se cumplen las invariantes, de manera que:
$i + 1 \leq k \leq N$
$pos = m : m \in [i \dots k)$ tal que $tmp = A[pos]$ y tmp es el valor menor en el subarreglo $A[i \dots k)$.
Se quiere mostrar que antes de la iteración $j = k + 1$ los invariantes siguen siendo ciertos si $A[k] \geq tmp$ se salta a la linea 9 y tmp y pos siguen siendo los mismos. En la linea 9:

```
j += 1
```

Con el valor $j = k$, $j = k + 1$.
Luego el invariantes I_2 se cumple pues al asumir que se pudo entrar en el ciclo, el valor máximo que pudo haber tomado k seria $N - 1$. $i + 1 \leq k + 1 \leq N$. De este modo al haber confirmado que $A[k]$ no era menor a nuestro tmp, aunque el rango del subarreglo se haya aumentado a $A[i \dots k + 1)$. Sabemos que se sigue cumpliendo que tmp es el valor menor. Y como tmp no cambio se mantiene que pos sea la posición de tmp en el arreglo A. Se cumplen I_3 e I_4.
Pero si en cambio $A[k] < tmp$ se ejecutan las lineas 7, 8 y luego 9:

```
tmp = A[j]
pos = j
j += 1
```

Con los valores $j = k$, entonces $j = k + 1$, $tmp = A[k]$ y $pos = k$.
Luego e invariante I_2 se cumple por el mismo análisis hecho para cuando $A[k] \geq$

tmp. Ahora el rango del subarreglo se ha aumentado en 1 así $A[i \dots k+1$, se supo que dentro de ese rango tmp no era el valor menor así que cambio a $A[k]$. Se sabe que es el menor es el arreglo pues fue menor a tmp y tmp cumplía que todos los demás elementos en el rango del subarreglo $A[0 \dots k]$ eran mayores o iguales a el. I_4 se cumple.
Ya que $tmp = A[k]$, pos tuvo que cambiar también. $pos = k$ corresponde con lo propuesto en I_3. También se cumple.
Finalmente el ciclo termina ya que el valor de j aumenta en 1 y eventualmente llegara a ser N, cumpliendo la condición de ciclo. Y así por la estabilidad de los invariantes tmp es el menor entre los elementos del subarreglo $A[i \dots N)$ y $pos \in [i \dots N]$ tal que $A[pos] = tmp$.
Teorema invariantes I_0, I_1. Los invariantes I_0, I_1 se cumplen.

Demostración: ...

3.4 Inicialización ($I_0 y I_1$)

Demostración: Se procede demostrar ...
De acuerdo a la linea 2:

```
i = 0
```

Si tomamos I_0: $0 \leq i \leq N \rightarrow 0 \leq 0 \leq N \rightarrow$ se cumple. Si tomamos I_1: El rango de $[0 \dots -1]$ representa un rango vació, trivialmente un subarreglo de 0 elementos esta ordenado. Por lo tanto se cumple.

3.5 Estabilidad ($I_0 y I_1$)

Se considera una iteración arbitraria $i = k$ y se asume que antes de ejecutar esta iteración se cumplen las invariantes: **A[0 .. k) está ordenado ascendentemente ...**
$0 \leq k \leq N$ y A esta ordenado ascendentemente hasta $k-1$. Se quiere mostrar que antes de la iteración $i = k+1$ las invariantes siguen ciertas. Al ejecutar las lineas 2 - 4:

```
i = 0
while i < len(l):
    j, tmp, pos = i + 1, l[i], i
    while j < len(l):
        if l[j] < tmp:
            tmp = l[j]
            pos = j
        j += 1
    l[i], l[pos] = tmp, l[i]
    i += 1
```

Luego se cumple I_0 pues al asumir que se pudo entrar en el ciclo el valor máximo que pudo haber tomado k era $N-1$. Por tanto $0 \leq k+1 \leq N$.
Así, se intercambia $A[k]$ con el valor menor del subarreglo $A[k \dots N]$. Y si todos los elementos en este subarreglo eran mayores o iguales a $A[k-1]$, entonces los elementos entre 0 e $i-1$ siendo $i = k+1$, siguen estando ordenados ascendente-

Se debería explicar bien el proceso e indicar el cumplimiento de los invariantes.

mente. Finalmente el ciclo termina ya que i empieza en 0 y va aumentando de 1 en 1. Eventualmente llegara a valer N y romperá la condición del ciclo. En ese punto por la estabilidad del invariante I_1 el arreglo $A[0 \dots i) = A[0 \dots N)$ estará ordenado ascendentemente. Por conservancia etc.

Teorema de correctitud: ...

4 Complejidad

4.1 Punto a: $6n^2 + 18n \in O(4n^2 logn)$

Demostración: Se procede a demostrar de forma directa. A partir de la definición de O es necesario mostrar que existen constantes n_0 y c tales que:

$$\forall_n \cdot n \geq n_0 \rightarrow 6n^2 + 18n \leq c \cdot 4n^2 logn$$

Luego se tiene que:
$6n^2 + 18n \leq c \cdot 4n^2 logn$
$\frac{6n^2 + 18n}{4n^2 logn} \leq c$
$\frac{3}{2logn} + \frac{9}{n^2 logn} \leq c$
$\frac{1}{logn}\left(\frac{3}{2} + \frac{9}{2n}\right) \leq c$
En este punto se asume que el valor mayor que puede tomar n es 2 pues $log_2 1 = 0$, $log_2 2 = 1$ y luego al ser $logn$ una función creciente. $\frac{1}{logn}$ tiende a 0.
Así con $n = 2$:
$\frac{1}{log_2 2\left(\frac{3}{2} + \frac{9}{2 \cdot 2}\right)} \leq c$
$1\left(\frac{3}{2} + \frac{9}{4}\right) \leq c$
$\frac{15}{4} \leq c$
Luego se puede concluir que $6n^2 + 18n \in O(4n^2 logn)$ con testigos $c = \frac{15}{4}$ y $n_0 = 2$.

4.2 Punto b: $2^{2n} \in O(2^n)$

Demostración: Se procede a refutar la afirmación. Para esto se procede por contradicción. Se asume que $2^{2n} \in O(2^n)$ y por ende tienen que existir constantes c y n_0 tales que $\forall_n \cdot n \geq n_0 \rightarrow 2^{2n} \leq c \cdot 2^n$. Luego se tiene que:
$2^{2n} \leq c \cdot 2^n$
$\frac{2^{2n}}{2^n} \leq c$
$2^{2n-n} \leq c$
$2^n \leq c$
No existe una constante c tal que $2^n \leq c$ ya que el valor de 2^n siempre puede ser mas grande, 2^n es una función creciente. Esto es una contradicción y por ende se concluye que $2^{2n} \notin O(2^n)$.

4.3 Punto c: $2^{n+2} \in O(2^n)$

Demostración: Se procede a demostrar de forma directa. A partir de la definición de O es necesario mostrar que existen constantes c y n_0 tales que $\forall_n \cdot n \geq n_0 \rightarrow 2^{n+2} \leq c \cdot 2^n$. Luego se tiene que:
$2^{n+2} \leq c \cdot 2^n$
$\frac{2^{n+2}}{2^n} \leq c$
$2^{n+2-n} \leq c$
$2^2 \leq c$
$4 \leq c$
Se toma $c = 4$ y como esta igualdad no depende de n se tiene que cumple para cualquier n. Se puede ver:
$2^{n+2} \leq 4 \cdot 2^n$
$2^{n+2} \leq 2^2 \cdot 2^n$
$2^{n+2} \leq 2^{n+2}$
$n + 2 \leq n + 2$
$n \leq n$
Luego se puede concluir que $2^{n+2} \in O(2^n)$ con testigos $c = 4$ y $n_0 = 1$.