

Tarea 2

Daniela Orobio 8977133

/ Nicolle Ruiz 8974036

Punto 1.

```
1 def algoritmo1(N):  
2     ans, ac, i = [], 1, 1  
3     while i <= N:  
4         ans.append(ac)  
5         ac = ac * (i + 1)  
6         i = i + 1  
7     return ans
```

Invariantes

I_0 $1 \leq i \leq N+1$

I_1 $1 \leq ac \leq (N+1)!$

I_2 $ans = [a_0, \dots, a_{i-2}]$

↳ tal que ans
contiene los $i-2$
factoriales en el
rango $[1, N]$.

con $N=5$

	i	ac	ans
Inicio	1	1	[]
	1	2	[1]
	2	6	[1, 2]
	3	24	[1, 2, 6]
	4	120	[1, 2, 6, 24]
	5	720	[1, 2, 6, 24, 120]
sale del ciclo	6	720	[1, 2, 6, 24, 120] ← return

este algoritmo calcula
Factoriales. Dado un N ,
se retorna un arreglo con
los factoriales de cada
numero de 1 a N .

Entrada: numero
entero N

($N > 0$) porque si
 $i \leq N$ ($i=1$) el ciclo
no iniciaria.

Salida: arreglo de
tamaño N con
los factoriales
de cada numero
entre 1 y N

Teorema: Las invariantes I_0 , I_1 y I_2 se cumplen

Demostración: Se procede mostrando la validez de los invariantes para la inicialización y la estabilidad.

Inicialización: de acuerdo con la línea 2 inicialmente $i = 1$, $ac = 1$ y $ans = []$

Para las invariantes I_0 y I_1 se tiene que:

$$1 \leq i \leq N+1$$

$$1 \leq ac \leq (N+1)!$$

$$1 \leq 1 \leq N+1 \quad \checkmark$$

$$1 \leq 1 \leq (N+1)! \quad \checkmark$$

Para el invariante I_2 se tiene que:

$$ans = [a_0, a_1, \dots, a_{i-2}] = [a_0, a_1, \dots, a_{1-2}]$$

$$ans = [a_0, \dots, a_{-1}]$$

$$ans = [] \quad \checkmark$$

dado que $[0, \dots, -1]$ es un rango vacío la lista ans estaría vacía, exactamente como como se inicia el arreglo ans en la línea 2. Además, de cumplirse que al inicio todavía no se han agregado elementos al arreglo.

por lo tanto, las invariantes I_0 , I_1 y I_2 se cumplen en la inicialización

Estabilidad: Se considera una iteración arbitraria en la que $i = K$ y $ac = h$ y se asume que antes de ejecutar esta iteración se cumplen los invariantes. En otras palabras, asumimos que:

$$1 \leq K \leq N+1 \quad \checkmark$$

$$1 \leq h \leq (N+1)! \quad \checkmark$$

$$ans = [a_0, a_1, \dots, a_{K-2}] \quad \checkmark$$

Son verdaderos. De modo que antes de esta iteración ans contiene los factoriales de los primeros K enteros en $[1, N]$. Ahora bien, es necesario mostrar después, cuando $i = i'$ y $ac = ac'$, las invariantes son ciertas y entonces:

$$1 \leq i' \leq N+1 \quad ?$$

$$1 \leq ac' \leq (N+1)! \quad ?$$

$$ans = [a_0, a_1, \dots, a_{i'-2}] \quad ?$$

deberían cumplirse. Al ejecutar las líneas 4-6 se tiene que:

$$\begin{array}{l} \text{(l\u00edneas)} \left\{ \begin{array}{l} 4 \text{ ans.append}(\underline{h}) \\ 5 \text{ ac} = h \cdot (K+1) \\ 6 \text{ i} = K+1 \end{array} \right. \end{array}$$

por lo que consecuentemente:

$$ans = [a_0, a_1, \dots, a_{k-2}, \underline{h}]$$

$$i' = i + 1 = k + 1$$

$$ac' = ac \cdot (i + 1) = h \cdot (k + 1)$$

por lo tanto las invariantes I_0 y I_1 continúan siendo verdaderas:

$$1 \leq i' \leq N + 1$$

$$1 \leq ac' \leq (N + 1)!$$

$$1 \leq k + 1 \leq N + 1$$

$$1 \leq h(k + 1) \leq (N + 1)!$$

puesto que en la línea 3 se tiene que $k \leq N$ y esto asegura $h(k + 1) < (N + 1)!$
 $k \leq N$ factorial de k factorial de N

y por esto también se cumple I_2 , ya que $h(k + 1)$ es el factorial de k , tal que $k \in [1, N]$.

Después, sabemos que el ciclo terminará cuando $i > N$, lo que terminará asegurando que:

$$1 \leq i \leq N + 1$$

$$1 \leq ac \leq (N + 1)!$$

$N + 1$

factorial de $N + 1$

$$ans = [a_1, a_0, \dots, a_{i-2}]$$

posición

$$i - 2 = (N + 1) - 2 = \underline{N - 1}$$

Como ans tiene los factoriales de $[1, N]$ la última posición de ans será $\underline{N - 1}$

Teorema: algoritmo $L(N)$ para cualquier entero N . ($N > 0$), produce una lista con los factoriales de los enteros en el rango $[2, N]$

Demostracion: es trivial a partir de la correctitud de los invariantes I_0 , I_1 y I_2


```

1 def algoritmo3(l):
2     i = 0
3     while i < len(l):
4         j, tmp, pos = i + 1, l[i], i
5         while j < len(l):
6             if l[j] < tmp:
7                 tmp = l[j]
9         j += 1
10        l[i], l[pos] = tmp, l[i]
11        i += 1

```

Entrada: Un arreglo $L[0 \dots N]$
de números $N \geq 0$

Salida: El arreglo L ordenado
ascendentemente

Invariantes

$I_0: 0 \leq i \leq N$ $I_1: \text{El arreglo } L[0 \dots i] \text{ ordenado ascendentemente}$

Para demostrar que los invariantes I_0, I_1 se cumplen es necesario poder establecer que el ciclo while interno realiza su trabajo apropiadamente. Para esto se hace un análisis de la invariante del ciclo

```

4         j, tmp, pos = i + 1, l[i], i
5         while j < len(l):
6             if l[j] < tmp:
7                 tmp = l[j]
9         j += 1

```

Invariantes:

$I_2: 1 \leq j \leq i+1$

$I_3: L[i] \leq tmp \leq L[N]$

$I_4: i \leq pos \leq N$

Teorema: Los invariantes I_2, I_3, I_4 se cumplen

Demostración: Se muestra la validez de los invariantes para la inicialización y estabilidad

Inicialización

De acuerdo a la línea 4

$$j = i+1, \text{tmp} = L[i], \text{pos} = i$$

$$1 \leq j \leq i+1$$

$$1 \leq i+1 \leq i+1$$

$$i \leq \text{pos} < N$$

$$0 \leq 0 < N \quad \checkmark$$

$$L[i] \leq \text{tmp} < N$$

$$L[0] \leq 0 < L(N) \quad \checkmark$$

Estabilidad: Se considera una iteración arbitraria en la que $j=k$ y se asume que antes de ejecutar esta iteración los invariantes son verdaderos.

$$1 \leq j \leq i+1$$

$$1 \leq k \leq i+1$$

```

1 vector<int> algoritmo2(int N){
2     vector<int> ans;
3     int i = 2;
4     while(N > 1){
5         if(N % i == 0){
6             ans.push_back(i);
7             N = N / i;
8         }
9         else
10            i += 1;
11     }
12     return ans;
13 }

```

Entrada: $0 \leq N \leq N'$

Salida:

$ans = [a_0, a_1, \dots, a_{K-1}] \vee \prod_{i=0}^{K-1} a_i = N'$
 Arreglo de los factores
 primo del numero N

Invariantes

$I_0: 0 \leq N \leq N'$

$I_1: 2 \leq i \leq N'$

$I_2: ans = [a_0, a_1, \dots, a_{K-1}]$ Donde K es el numero de factores primos de un numero N

Donde los K valores de N pertenecen a los factores primo

$$\prod_{i=0}^{K-1} a_i = N'$$

$N' \rightarrow$ Donde N' es el
numero inicial de N

Teorema: Las invariantes I_0, I_1, I_2 se cumplen

Demostración: Se produce mostrando la validez de los invariantes para la inicialización y estabilidad

Inicialización

\rightarrow De acuerdo a los lineas 1, 2, 3 $\rightarrow N=N, i=2, ans=empty$

Para la invariante I_0 se tiene que

Se llama un entero N al algoritmo que debe ser ≥ 0
para que el algoritmo funcione

$$\begin{aligned} 0 \leq N \leq N \\ 0 \leq 0 \leq 0 \quad \checkmark \end{aligned}$$

Para la invariante I_1 Se tiene que

$$\begin{aligned} 2 \leq i \leq N \\ 2 \leq 2 \leq N \end{aligned}$$

Para la invariante I_2 Se tiene que

$$ans = [a_0, a_1, \dots, a_{k-1}] = [a_0, a_1, \dots, a_{0-1}]$$

$$ans = [a_0, \dots, a_{-1}]$$

$$ans = []$$

$$\prod_{j=0}^{0-1} a_j = N \rightarrow \prod_{i=0}^{-1} a_i = 0$$

↑
la suma
no tiene ningún
elemento

Dado que el rango entre $[a_0, a_{-1}]$ es un rango vacío
la lista debe estar vacía. Además se satisface que
 ans contiene los factores primos de un N , y como
1 no tiene factores primos entonces debe retornar
un arreglo vacío

Por lo tanto los invariantes I_0, I_1, I_2 se cumplen
en la inicialización

Estabilidad

Se considera una iteración arbitraria en la que $i=m$ y $N=x$ y se asume que antes de ejecutar esa iteración se cumplen los invariantes. Eso quiere decir que

$$2 \leq m \leq x' \quad \checkmark \quad \text{y} \quad 1 \leq x \leq x' \quad \checkmark$$

$$a[n] = [a_0, a_1, \dots, a_{K-1}] \quad \checkmark \quad \left(\prod_{j=0}^{K-1} a_j = N' \right) \quad \checkmark$$

Ahora el objetivo es demostrar que después de esta iteración y donde N es un número > 1 son Verdaderos

$$2 \leq m \leq x' \quad ? \quad \text{y} \quad 1 \leq x \leq x' \quad ?$$

$$a[n] = [a_0, a_1, \dots, a_{K-1}] \quad \left(\prod_{j=0}^{K-1} a_j = N' \right) \quad ?$$

Al ejecutar las líneas 5-10 se tienen las siguientes posibilidades

- La condición de la línea 5 es verdadera por lo que se tiene que $x \% m \neq 0$, por lo que al ser verdadero

Se ejecutan los siguientes lineas 6-7

$$\begin{array}{l} \text{ano.append}(m) \\ x = x/m \end{array}$$

En consecuencia

$$\text{ano} = [a_0, a_1, \dots, a_{k-1}, m]$$
$$x = x/m \quad m = m$$

De esta forma, los invariantes I_0 y I_1 , continúan siendo Verdadero

$$\begin{array}{l} 2 \leq m \leq N' \\ 2 \leq i \leq N' \end{array}$$
$$\begin{array}{l} 0 \leq x \leq N' \\ 0 < N/i \leq N' \end{array}$$

De esta misma forma se repetira el mismo proceso hasta que $x \cdot i \neq 0$ es decir que la linea 5 se hace falsa o que $N < 1$ donde ya no entra en a ningun ciclo y se acabaria el programa

En caso de que $x \cdot i \neq 0$ se ejecuta la linea 9-10

$$m += 1$$

De esta forma, los invariantes I_0 y I_1 , continúan siendo Verdadero

$$\begin{array}{l} 2 \leq m \leq N' \\ 2 \leq i+1 \leq N' \end{array}$$
$$\begin{array}{l} 0 \leq x \leq N' \\ 0 \leq N \leq N' \end{array}$$

Siendo así ans tendrá todos los factores primos del número

Teorema: La invocación $algoritmo2(N)$ para cualquier N produce un arreglo de todos los números primos del número

Demostración: Es trivial a partir de la correctitud de los invariantes $I_0, I_1, I_2,$

Punto 4.

definición de O

Sea $f: \mathbb{N} \rightarrow \mathbb{R}_{>0}$ se tiene que:

$$O(f) = \{g: \mathbb{N} \rightarrow \mathbb{R}_{>0} \mid \exists n_0 \in \mathbb{N}. \exists c \in \mathbb{R}_{>0}. \forall n \in \mathbb{N}. n \geq n_0 \rightarrow g(n) \leq c \cdot f(n)\}$$

a) $6n^2 + 18n \in O(4n^2 \log n)$

↳ por def se tienen n_0 y c

$$\frac{6n^2 + 18n}{4n^2} \leq c \cdot \frac{4n^2 \log n}{4n^2} \quad \forall n. n \geq n_0$$

$$\frac{\frac{3}{2} + \frac{9}{2n}}{\log n} \leq c \frac{\log n}{\log n} \quad \text{simplificar}$$

entonces asumiendo que $n_0 = 2$ y $c = 4$

↳ sustituyendo n

$$\frac{\frac{3}{2} + \frac{9}{2(2)}}{\log(2)} \leq 4$$

$$\frac{\frac{6+9}{4}}{2} \leq 4$$

$$\frac{15}{4} \leq 4$$

$$\frac{15}{4} \approx 3,75$$

$$3,75 \leq 4 \quad \text{se cumple}$$

entonces $6n^2 + 18n \in O(4n^2 \log n)$ con $n_0 = 2$ y $c = 4$

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

↳ por def se tienen n_0 y c

$$2^{2n} \leq c \cdot 2^n \quad \forall n. n \geq n_0$$

$$2^n \cdot 2^n \leq C \cdot 2^n \quad a^{m+n} = a^m \cdot a^n$$

$$\frac{2^n \cdot 2^n}{2^n} \leq \frac{C \cdot 2^n}{2^n} \quad \text{simplificar}$$

$$2^n \leq C$$

entre mayor sea n mayor es su resultado

Lo y ya que el lado derecho es constante, siempre podran existir un n que rompa la desigualdad, por lo tanto no se cumple

entonces $2^{2n} \notin O(2^n)$

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

Lo por def se tiene n_0 y C

$$2^{n+2} \leq C \cdot 2^n \quad \forall n. n \geq n_0$$

$$2^n \cdot 2^2 \leq C \cdot 2^n \quad a^m \cdot a^n = a^{m+n}$$

$$\frac{2^n \cdot 2^2}{2^n} \leq \frac{C \cdot 2^n}{2^n} \quad \text{simplificar}$$

$$4 \leq C \quad \text{son constantes}$$

entonces asumiendo $n_0 = 1$ y $C = 4$

$$2^{(1)+2} \leq (4) 2^{(1)}$$

$$2^3 \leq 8$$

$$8 \leq 8 \quad \text{se cumple}$$

entonces $2^{n+2} \in O(2^n)$ con $n_0 = 1$ y $C = 4$