Tarea 2

Punto 1.

```python
def algoritmo1(N):
  ans, ac, i = [], 1, 1
  while i <= N:
    ans.append(ac)
    ac = ac * (i + 1)
    i = i + 1
  return ans
```

Invariantes

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

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

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

↳ tal que ans contiene los i-1 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 <= 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 linea 2 inicialmente $i = 1$, $ac = 1$ y $ans = [\,]$

Para las invariantes I_0 y I_1 se tiene que:

$$1 \le i \le N+1 \qquad 1 \le ac \le (N+1)!$$

$$1 \le 1 \le N+1 \checkmark \qquad 1 \le 1 \le (N+1)! \checkmark$$

Para el invariante I_2 se tiene que:

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

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

$$ans = [\,] \checkmark$$

dado que $[0, \dots, -1]$ es un rango vacio la lista ans estaria vacia, exactamente como como se inicia el arreglo ans en la linea 2. Ademas de cumplirse que al inicio todavia no se han agregado elementos al arreglo.

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

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:

$1 \leq K \leq N+1$ ✓ $\quad 1 \leq h \leq (N+1)!$ ✓

$ans = [a_0, a_1, \ldots, a_{K-2}]$ ✓

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:

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

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

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

líneas
4 ans.append(h)
5 $ac = h \cdot (K+1)$
6 $i = K+1$

por lo que consecuentemente:

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

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

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

por lo tanto las invariantes I_0 y I_1 continuan siendo verdaderos:

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

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

puesto que en la linea 3 se tiene que $k \leq N$ y esto asegura $\underbrace{h(k+1)}_{\text{factorial de } k} < \underbrace{(N+1)!}_{\text{factorial de } N}$

$k < N$

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

Despues, sabemos que el ciclo terminara cuando $i > N$, lo que terminara asegurando que:

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

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 ultima posición de ans sería $\underline{N-1}$

Teorema: algoritmo 1 (N) para cualquier entero N. (N > 0), produce una lista ans con los factoriales de los enteros en el rango [1, N]

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

```python
def algoritmo3(l):
  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]
          j += 1
      l[i], l[pos] = tmp, l[i]
      i += 1
```

Entrada: Un arreglo $L[0 \ldots N]$ de numeros $N \geq 0$

Salida: El arreglo L ordenado ascendentemente

Invariantes

$I_0: 0 \leq i \leq N$ I_1: El arreglo $L[0 \ldots i]$ 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 analisis de la invariante del ciclo

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

Demostracion: Se muestra la validez de los invariantes para la inicializacion y estabilidad

Inicialización

De acuerdo a la línea 4

$j = i+1$, $tmp = L[i]$, $pos = i$

$1 \leq j \leq i+1$ $\qquad$ $i \leq pos < N$ $\qquad$ $L[i] \leq tmp < N$

$1 \leq 1+i \leq i+1$ $\qquad$ $0 \leq 0 < N$ ✓ $\qquad$ $L[0] \leq 0 < L(N)$ ✓

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:

$ans = [a_0, a_1, \ldots a_{K-1}] \vee \prod_{i=0}^{K-1} a_i = N'$

→ Arreglo de los factores primos del numero N

Invariantes

N' → Donde N' es el numero inicial de N

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

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

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

Donde los K valores de N pertenecen a los factores primos

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

Teorema: Las invariantes I_0, I_1, I_2 se cumplen

Demostracion: Se produce mostrando la validez de los invariantes para la inicializacion y estabilidad

Inicializacion

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

Para la invariante I_0 se tiene que

Se llama un entero N al algoritmo que debe ser $\geq \emptyset$ para que el algoritmo funcione

$\emptyset \leq N \leq N'$

$\emptyset \leq \emptyset \leq \emptyset$ ✓

Para la invariante I_1 se tiene que

$2 \leq i \leq N'$

$2 \leq 2 \leq N'$

Para la invariante I_2 se tiene que

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

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

$\prod_{j=0}^{0-1} a_j = N' \Rightarrow \prod_{i=0}^{-1} a_0 = \emptyset$

La sumatoria no tiene ningun elemento

$ans = [\,]$

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

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

Estabilidad

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

$$2 \leq m \leq x' \checkmark \quad y \quad 1 \leq x \leq x' \checkmark$$

$$ans = [a_0, a_1, \ldots a_{K-1}] \checkmark \quad \left(\prod_{J=0}^{K-1} a_J = N'\right) \checkmark$$

Ahora el objetivo es demostrar que despues de esta iteracion y donde N es un numero >1 son verdaderos

$$2 \leq m \leq x' \ ? \quad y \quad 1 \leq x \leq x' \ ?$$

$$ans = [a_0, a_1, \ldots a_{K-1}] \quad \left(\prod_{J=0}^{K-1} a_J = N'\right) ?$$

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

- La condicion de la linea 5 es verdadera por lo que se tiene que $X \% m == 0$, por lo que al ser verdado

Se ejecutan los siguientes lineas 6-7

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

En consecuencia

$$ans = [a_0, a_1, \ldots a_{k-1}, m] \qquad \boxed{x = x/m \quad m = m}$$

De esta forma, los invariantes I_0 y I_1, continuan siendo verdadero

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

De esta misma forma se repetira el mismo proceso hasta que $x \% i != 0$ es decir que la linea 5 se hace falsa o que $N < 1$ donde ya no entrona a ningun ciclo y se acaboria el programa

En caso de que $x \% i != 0$ se ejecuta la line 9-10

$$\boxed{m += 1}$$

De esta forma, los invariantes I_0 y I_1, continuan siendo verdadero

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

Siendo asi ans tendra todos los factores primos del numero

Teorema: La invocacion algoritmo 2(N) para cualquier N produce un arreglo de todos los numeros primos del numero

Demostracion: Es trivial a partir de la correctitud de los invariantes I_0, I_1, I_2 //

Punto 4.

definición de O

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

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

---

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

↳ por def se tienen n_0 y c

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

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

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

↳ sustituyendo n

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

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

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

$$3,75 \leq 4$$ 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

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

$$2^n \cdot 2^n \leq c \cdot 2^n \qquad a^{m+n} = a^m \cdot a^n$$

$$\frac{2^n \cdot \cancel{2^n}}{\cancel{2^n}} \leq \frac{c \cdot \cancel{2^n}}{\cancel{2^n}} \qquad \text{simplificar}$$

$$2^n \leq c$$

entre mayor sea n mayor es su resultado

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

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

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

por def se tiene n_0 y c

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

$$2^n \cdot 2^2 \leq c \cdot 2^n \qquad a^m \cdot a^n = a^{m+n}$$

$$\frac{\cancel{2^n} \cdot 2^2}{\cancel{2^n}} \leq \frac{c \cdot \cancel{2^n}}{\cancel{2^n}} \qquad \text{simplificar}$$

$$4 \leq c$$ son constantes

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

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

$$2^3 \leq 8$$

$$8 \leq 8$$ se cumple

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