Tarea 6
Algoritmos

Emmanuel Peto Gutiérrez
José Luis Vázquez Lázaro

21 de noviembre de 2022

Problema 2

a)
Sea $long(i)$ la longitud de la subsecuencia creciente más larga tal que el último elemento de esa subsecuencia es s_i, con $0 \leq i \leq n-1$. Entonces se define la función $long(i)$ de la siguiente manera:

1) $long(0) = 1$

2) $long(i) = 1$ si ningún elemento a la izquierda de i es menor que s_i.

3) $long(i) = 1 + \text{máx}(long(j))$ con $j < i$ y $s_j < s_i$, en otro caso.

Demostración por inducción sobre i.
Caso base: $i = 0$.
Por definición, $long(0) = 1$, lo que significa que una subsecuencia creciente que empiece en el índice 0 y termine en el índice 0 tiene exactamente un elemento, lo cual es correcto.
H.I. Sea $i > 0$ y supongamos que para cualquier $j < i$ se cumple que $long(j)$ es la longitud de la subsecuencia creciente más larga cuyo último elemento es s_j.
Se demostrará que $long(i)$ es la longitud de la subsecuencia creciente más larga cuyo último elemento es s_i.
Caso 1: ningún elemento a la izquierda de i es menor que s_i.
En este caso, la subsecuencia creciente más larga (a partir de este punto se abreviará como SCML) sólo puede contener un elemento, el cual es s_i, pues todos los elementos a la izquierda son mayores o iguales que s_i. Para este caso se define $long(i) = 1$, y por lo tanto es correcto.
Caso 2: existe al menos un elemento a la izquierda de i que es menor que s_i.
Sea S' la SCML construída solo con elementos de $\{s_0, s_1, ..., s_{i-1}\}$ con la condición de que el último elemento de esa subsecuencia es menor que s_i, digamos que ese elemento es s_k. Se puede construir una SCML con elementos de

$\{s_0, ..., s_i\}$ simplemente agregando s_i al final de S' ($S' + +[s_i]$), y la longitud de esa SCML es la longitud de S' más 1.

Por H.I. $long(k) = |S'|$, pues $k < i$. Se está suponiendo que S' es la subsecuencia más larga con el último elemento menor que s_i de forma que S' sólo contiene elementos que están a la izquierda de i, así que $\max(long(j)) = long(k) = |S'|$, donde $j < i$ y $s_j < s_i$. Usando estas igualdades y la definición de $long(i)$, se obtiene que $long(i) = |S'| + 1$, y por lo tanto $long(i)$ es la longitud de la SCML construída solo con elementos de $\{s_0, ..., s_i\}$ de forma que s_i es el último elemento. ■

b)

Se describe el algoritmo en los siguientes pasos:

1. Construir un arreglo L de tamaño n donde todas sus entradas tienen 0.
2. Hacer $L[0] = 1$.
3. Para cada i en $\{0, ..., n-1\}$, calcular recursivamente longitud de la SCML cuyo último elemento es s_i y guardar el resultado en $L[i]$ (usando la ecuación de Bellman para este problema).
4. Devolver el elemento más grande del arreglo L.

Las siguientes funciones en python calculan la longitud de la SCML. La función *maximaSub* calcula recursivamente la longitud de la SCML cuyo último elemento es $S[indice]$, mientras que la función *scml* calcula la longitud de la SCML del arreglo completo S.

```
1: def maximaSub(S, L, i):
2:     if L[i] == 0:
3:         maxLon = 0
4:         for j in range(i):
5:             temp = maximaSub(S, L, j)
6:             if (temp > maxLon) and (S[j] < S[i]):
7:                 maxLon = temp
8:         L[i] = maxLon + 1
9:     return L[i]
```

```
1: def scml(S):
2:     L = [0 for elem in S]
3:     L[0] = 1
4:     maximaSub(S, L, len(S)-1)
5:     return max(L)
```

<u>Correctitud.</u>

Para comprobar que es correcto se verá que $L[i]$ guarda el valor de la longitud de la SCML con último elemento $S[i]$. Si $i = 0$ entonces $L[i]$ tiene que ser 1, lo cual es correcto por la línea 3 de *scml*. Si $i > 0$ entonces $L[i]$ tiene que ser $\max(maximaSub(j)) + 1$, con $j < i$ y $S[j] < S[i]$. Entre las líneas 3 y 7 de *maximaSub* se encuentra al elemento $\max(maximaSub(j))$ que cumple con la condición $S[j] < S[i]$ (por la línea 6) y se guarda en la variable $maxLon$. En la línea 8 se calcula finalmente el valor de $L[i]$, el cual será $maxLon + 1$, y se devuelve ese valor calculado. Observe que si ningún elemento a la izquierda de i es menor que $S[i]$, entonces $maxLon$ se quedará en 0 y $L[i]$ será 1.

Complejidad.

Tómese un índice fijo i. Suponiendo que ya se calcularon los valores de $L[j]$, para toda $j < i$, encontrar al máximo $L[j]$ toma tiempo $O(n)$. Una vez que se tiene a ese máximo, calcular el valor de $L[i]$ y devolverlo toma tiempo $O(1)$. Luego, este proceso se tiene que repetir n veces para toda $i \in 0, ..., n-1$ y el valor $L[i]$ se calcula exactamente una vez (cuando se comprueba que $L[i]$ es 0), por lo que la complejidad total es $O(n^2)$.

c)

En las siguientes funciones de python, *scmlIt* y *maximaSubIt*, se obtienen las versiones iterativas de las funciones *scml* y *maximaSub*.

```python
def maximaSubIt(S, L):
    L[0] = 1
    for i in range(1, len(S)):
        maxLon = 0
        for j in range(i):
            temp = L[j]
            if (temp > maxLon) and (S[j] < S[i]):
                maxLon = temp
        L[i] = maxLon + 1
```

```python
def scmlIt(S):
    L = [0 for elem in S]
    L[0] = 1
    maximaSubIt(S, L)
    return max(L)
```

La correctitud y la complejidad son inmediatas del inciso anterior.

d)

Ahora supongamos que la función *scmlIt* no devuelve el máximo elemento de L sino que devuelve toda la lista L. Se va a construir una lista llamada *salida* que va a contener los elementos de la SCML. Los pasos del algoritmo se describen a continuación.

1. Obtener la lista L con la función *scmlIt*.

2. Crear una lista (*salida*) con un solo elemento, el cual será el primer elemento de la secuencia S.

3. Crear una variable entera (*longAct*) con valor 2. Esta variable indica el tamaño de la lista que se quiere construir hasta la iteración actual.

4. Para cada i desde 1 hasta $n-1$, hacer lo siguiente:

4.1 Si $L[i]$ es igual a $longAct - 1$ y $S[i]$ es menor al último elemento de *salida*, entonces sustituir a ese último elemento por $S[i]$.

4.2 Si $L[i]$ es igual a $longAct$ y $S[i]$ es mayor al último elemento de *salida*, agregar $S[i]$ al final de *salida* e incrementar en 1 el valor de $longAct$.

El siguiente código en python construye esa lista.

```python
def construyeSCML(S):
    L = scmlIt(S)
    salida = []
    salida.append(S[0])
    longAct = 2
    for i in range(1, len(L)):
        if (L[i] == longAct-1) and (S[i] < salida[len(salida)-1]):
            salida[len(salida)-1] = S[i]
        elif (L[i] == longAct) and (S[i] > salida[len(salida)-1]):
            salida.append(S[i])
            longAct += 1
    return salida
```