AyC 2023 - Actividad 7- Jueves 4/5/2023 - Grupo de trabajo (Indicar Apellidos/Nombres):

Supongamos que tenemos una secuencia de números, donde cada elemento representa el número máximo de pasos que se pueden dar hacia adelante para avanzar en la secuencia a partir de esa posición.
Ejemplo: si tenemos la secuencia 2 1 2 3 1 1 1 1 significa que si estamos en la primera posición (que tiene un 2) podemos avanzar uno o dos pasos hacia adelante. Pero, si estamos en la quinta posición (donde hay un 1) solo podemos avanzar una posición hacia adelante.

El problema a resolver consiste en encontrar cuál es el mínimo número de pasos necesarios para avanzar desde el inicio de la secuencia hasta el final.

En la secuencia de nuestro ejemplo, necesitamos cuatro pasos: **2** 1 **2 3** 1 1 **1 1**
(se marcan en negrita/subrayado las posiciones que se visitan con los pasos dados.)
Observemos que podríamos haber realizado otra secuencia de pasos, por ejemplo **2 1 2** 3 **1 1 1 1**, pero tiene más pasos.

Propuesta: encontrar una solución de PD para resolver el problema.

Consignas a resolver

0,50 a) -defina una función saltar(i) que modele el mínimo número de saltos necesarios para avanzar en la secuencia dada desde la posición i hasta el final. (1pto diseño)

1,75 b) Explique claramente qué estructuras de datos podría utilizar en un algoritmo para resolver el problema utilizando la función del inciso anterior. Indique claramente cómo las tendría que inicializar, recorrer y actualizar a fin de resolver el problema (2ptos ED).

0.00 c) - Dar un algoritmo que resuelva el problema utilizando la definición dada en a) y las ED indicadas en b) (1 pto diseño)

0.50 d) Analice el tiempo de ejecución y el espacio requerido por el algoritmo dado. (2ptos A/C)

2.00 e) -¿Qué dice el principio de optimalidad? ¿Se cumple en este problema? Justifique.
- Explique el concepto de superposición de subproblemas, muestre un ejemplo particular del mismo en este problema. (2ptos Apl/uso).

... permite reutilizar los resultados previamente calculados

a)

$$saltar(i) = \begin{cases} 0 & \text{si } i = n \\ 1 + \min(saltar(i+k)) & \text{si } i < n \end{cases}$$

↓ ¿k? → x

siendo n la longitud de la secuencia y k el valor de la posición i

↓

eso es: solo un valor? ¿

b) Podríamos utilizar una estructura de datos lineal donde la inicializamos con -1 en todos sus elementos, el tamaño de la estructura será igual al tamaño de la secuencia (n), lo recorremos de izquierda a derecha almacenando el resultado de la función saltar(i) lo que nos permite reutilizar los resultados previamente calculados y mejorar la eficiencia del algoritmo

→ calcula Si antes que Sj si i<j? no coincide c/ lo indicado en (a)

c)

(al final)

e) El principio de optimalidad es un principio de la programación dinámica y establece que cualquier subsecuencia de decisiones que tenga el mismo estado final debe ser óptima respecto al subproblema correspondiente, lo que significa que

la solución óptima a un problema se puede obtener dividiendolo en subproblemas más pequeños y resolver cada subproblema de manera óptima. Si lo cumple porque al dividir en subproblema cada subproblema genera el mínimo. ¿genera?

La superposición de subproblemas se refiere a la situación en la que un problema se puede dividir en subproblemas que se resuelven varias veces. Acá lo usamos cuando tratamos de calcular el mínimo número de pasos desde diferentes posiciones de la secuencia. Por ejemplo si la secuencia es [2, 1, 2, 3, 1, 1] al calcular el min número de pasos desde la posición 1 podemos considerar saltar a la posición 1 o 2. Sin embargo al calcular el min num de pasos desde la posición 2 también necesitamos considerar saltar a la posición 2, acá el subproblema de encontrar el mínimo número de pasos desde la posición 2 (1) se repite en ambos casos lo que demuestra la superposición de problema.

✓

c)

```
min Jump ( secuencia , n )
    Jump [n]
    for i=1 to n
        Jump [i] := -1
    min = MAX_INT
    for i=1 to n
        for j=1 to (j<i) ?
            If ( min > jump [j] )
                min := jump[j]
    If ( min != MAX_INT )
        Jump[i] := min + 1
    else
        Jump[i] := min
    Return [i] x
```

¿semántico?

no usa ningún valor de la secuencia.

→ sigue la recurrencia dada en (2)?

¿i? → i es la variable de control de la estructura "for".

d) El tiempo de ejecución es $O(n^2)$, ya que el recorrido anidado de la matriz

El espacio utilizado es de $O(n)$ para almacenar el array de resultados de valor

¿qué matriz?