

Tarea 2

Análisis y Diseño de Algoritmos

Estudiante:

Juan Diego Valencia Alomia

Código:

8977467

Docentes:

Camilo Rocha & Carlos Ramírez

Santiago de Cali, febrero 26, 2025

Punto 1)

Explicación del problema

El problema trata sobre maximizar el salario

con pagos semanales en N semanas. Se puede

hacer trabajos de bajo estrés y de alto estrés.

No obstante si deseas trabajar de alto estrés la

semana anterior no debiste haber trabajado

- (a) Show that the following algorithm does not correctly solve this problem, by giving an instance on which it does not return the correct answer.

```
For iterations  $i = 1$  to  $n$ 
  If  $h_{i+1} > \ell_i + \ell_{i+1}$  then
    Output "Choose no job in week  $i$ "
    Output "Choose a high-stress job in week  $i+1$ "
    Continue with iteration  $i+2$ 
  Else
    Output "Choose a low-stress job in week  $i$ "
    Continue with iteration  $i+1$ 
  Endif
End
```

To avoid problems with overflowing array bounds, we define $h_i = \ell_i = 0$ when $i > n$.

In your example, say what the correct answer is and also what the above algorithm finds.

a)

Contra ejemplo:

Sea $L = [5, 5, 5]$ y $H = [50, 0, 35]$

Para $i = 1$:

Verificar $h_2 > \ell_1 + \ell_2$, como $h_2 = 0$, $\ell_1 = 5$ y $\ell_2 = 5$, Entonces no se cumple, ya que $0 < 10$ y escoge hacer low stress, nuestro acumulado es 5

Para $i = 2$:

Verificar $h_3 > \ell_2 + \ell_3$, como $h_3 = 35$, $\ell_2 = 5$ y $\ell_3 = 5$, Entonces se cumple, ya que $35 > 10$ y escoge hacer high, el algoritmo finaliza y da que la solución es 40

Este resultado es incorrecto ya que la solución óptima es hacer High stress

en la semana 1, None en la semana 2 y High stress en la semana 3, lo

que daría un total de $50 + 35 = 85$

b)

Especificación del problema

- Entrada: Dos arreglo $L[0...N]$ y $H[0...N]$ de enteros positivos, $N \geq 0$ indicando los salarios semanales de trabajo de poco estrés y alto estrés respectivamente

- Salida: El salario máximo que resulta después de N semanas, según las condiciones

Planteamiento de la función objetivo

Sea $0 \leq i \leq N$

$\phi(i)$: calcula el salario maximo en N semanas

Reformulación Especificación

Salida: $\phi(N)$

Planteamiento recursivo

$$\phi(i) = \begin{cases} 0 & \text{if } i = 0 \\ \max(L[i-1], H[i-1]) & \text{if } i = 1 \\ \max(L[i-1] + \phi(i-1), H[i-1] + \phi(i-2)) & \text{if } i > 1 \end{cases}$$

Código:

```
def solve(i):  
    Sea dp  
    un diccionario  
    de python  
    if i == 0:  
        ans = 0  
    elif i == 1:  
        ans = max(L[i-1], H[i-1])  
    elif i in dp:  
        ans = dp[i]  
    else:  
        ans = max(L[i-1] + solve(i-1), H[i] + solve(i-2))  
        dp[i] = ans  
    return ans
```

Correctitud

Teorema:

Sea N la cantidad total de elementos, $N = \text{len}(L) = \text{len}(H)$,

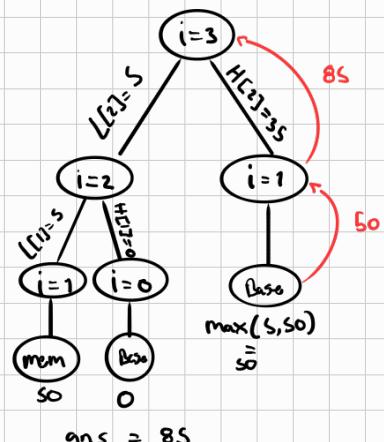
función $\phi(i)$ con $0 \leq i \leq N$, retorna la suma maxima que se puede obtener

Combinando los elementos de los arreglos L y H según las siguientes condiciones de la recursividad

Ejemplo

Sea $L = [5, 5, 5]$ y

$H = [50, 0, 35]$



Caso Base:

1) $i = 0$

$\phi(0)$ retorna 0, ya que no se trabajó ninguna semana, por lo tanto es correcto

2) $i = 1$

$\phi(1)$ retorna el $\max(L[0], H[0])$ que es la suma máxima

posible cuando se trabajó solo 1 semana, por lo tanto es correcto

Hipótesis inductiva

Existe un $n, k \in \mathbb{N}$, tal que $0 \leq k \leq i$, entonces $\phi(k)$ retorna

la suma máxima correcta combinando los elementos de L y H

Paso inductivo:

Se quiere demostrar que $\phi(i)$ retorna correctamente la suma máxima posible $\forall i \geq 1$

lo que nos lleva a dos casos

1) Se escoge $L[i-1]$ y se suma $\phi(i-1)$

En efecto, por H.I se sabe que $\phi(i-1)$ retorna una solución correcta

Por lo que este caso es válido

2) Se escoge $H[i-1]$ y se suma $\phi(i-1)$

De nuevo, por H.I se sabe que $\phi(i-1)$ da la suma

máxima para los primeros $i-1$ elementos, así que este opción es válida

Dado que para $i \geq 1$ se escoge uno de los dos casos, que por hipótesis, ya son correctos, entonces $\phi(i)$ también retorna la solución óptima

Ánalisis de complejidad

Dado al uso de la técnica de programación dinámica, la memorización se realiza en $O(N)$,

ya que cada valor de $\phi(i)$ se calcula una única vez y se almacena en el diccionario dp

Punto 2)

Explicación del problema

Se quiere destruir la máxima cantidad de robots con un arma. Estos robots llegan por oleadas, cada una determinada por i segundos, además el arma se carga al pasar el tiempo, pero se pierde una vez usada.

a) Contraseña

Sea $X = [6, 6]$ y $f[4, 6]$ para $N=2$

Según el programa se debe verificar el j más pequeño para el cual se cumpla que

$$f(j) \geq X_n$$

En este caso $X_1 = X_2 = 6$, por lo tanto el j más pequeño es $j=2$ donde $f(2) = 6$, y se cumple que

$$6 \geq 6$$

Por lo que se activa el arma en el segundo $j=2$ y se determina la cantidad de robots en este caso sería:

$$\min(6, 6) = 6$$

De esta manera procede a hacerse la verificación

$$n-j \geq 1$$

y con $n=2 \wedge j=2$, entonces no se cumple la condición ya que

$$0 < 1$$

De esta manera el programa para y retorna que la suma máxima es 5,

sin embargo esto es erróneo ya que el camino óptimo es activar el arma en $j=1$

por lo que la suma se actualiza a $\min(6, 4) = 4$, y luego en el segundo $j=2$ activar nuevamente el arma, por lo tanto $\text{sum} = 4 + \min(6, 4) = 4 + 4 = 8$

Por lo que la suma máxima real sería 8 robots

- (a) Show that the following algorithm does not correctly solve this problem, by giving an instance on which it does not return the correct answer.

Schedule-EMP(x_1, \dots, x_n)

Let j be the smallest number for which $f(j) \geq x_n$

(If no such j exists then set $j = n$)

Activate the EMP in the n^{th} second

If $n-j \geq 1$ then

Continue recursively on the input x_1, \dots, x_{n-j}
(i.e., invoke Schedule-EMP(x_1, \dots, x_{n-j}))

In your example, say what the correct answer is and also what the algorithm above finds.

Punto 2.b)

Especificación de problema

Entrada: Dos Arreglos $x[0 \dots N]$ y $f[0 \dots N]$ de números enteros positivos, $N \geq 0$, indicando la llegada de robots el el tiempo x_i y la función de potencia en el tiempo $f(i)$ respectivamente

Salida: un entero positivo que representa la cantidad máxima de robots que se pueden destruir al activar el arma

Planteamiento de la función objetivo

Sea $0 \leq i \leq N \wedge 0 \leq j \leq N$

$\phi(i, j)$: Calcular la Maxima cantidad de robots que se pueden destruir

Reformulación Salida

$\phi(0, 0)$

Planteamiento recursivo

$$\begin{cases} 0 & \text{if } i = N \\ \max(\min(x[i], f[j]) + \phi(i+1, 0), \phi(i+1, j+1)) & \text{if } i < N \end{cases}$$

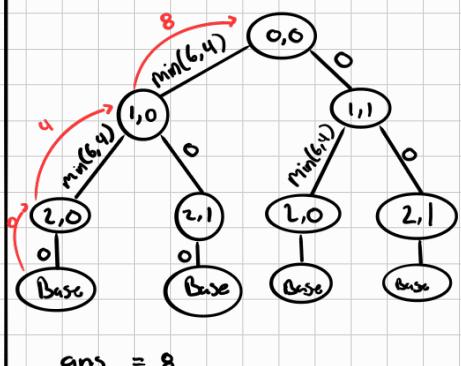
Código

Sea dp un diccionario global de Python, entonces

```
def solve(i, j):  
    ans = None  
    if i == N:  
        ans = 0  
    elif (i, j) in dp:  
        ans = dp[i, j]  
    else:  
        ans = max(min(x[i], f[j]) + solve(i+1, 0), solve(i+1, j+1))  
    dp[i, j] = ans  
    return ans
```

Ejemplo

Sea $x = [6, 6]$ y $f = [4, 6]$



Correctitud:

Teorema

Sea $X[0\dots N]$ y $f[0\dots N]$ arreglos de enteros, $N \in \mathbb{N}$ y $N \geq 0$, la función $\text{solve}(i)$, ($0 \leq i \leq N$), retorna correctamente la suma máxima de robots que se pueden destruir después de i segundos.

Se procede por medio de inducción matemática sobre el tamaño del arreglo X y f .

Caso Base

$i = N$

En este caso ya se procesaron todos los elementos de los arreglos X y f .

Por lo que la suma máxima será 0, ya que no hay elementos que procesar.

Paso inductivo

Se quiere probar que $\text{solve}(i, j)$ retorna la suma máxima en el intervalo $[i \dots N]$.

Hipótesis inductiva

Se establece que $\text{solve}(i+1, 0)$ y $\text{solve}(i+1, j+1)$ retornan correctamente los valores óptimos para los subproblemas en el rango $[i+1 \dots N]$.

En efecto, para esta instancia se derivan dos casos:

1) Tomar el $\min(X[i], f[j])$ y continuar con $\text{solve}(i+1, 0)$ reiniendo el arma.

Por H.I se sabe que $\text{solve}(i+1, 0)$ es correcto.

2) No disparar y continuar con $\text{solve}(i+1, j+1)$.

Por H.I se sabe que $\text{solve}(i+1, j+1)$.

Ahora bien, $\max()$ puede escoger entre el caso 1 y 2, y como se sabe que son correctos, entonces $\text{solve}(i, j)$ con $i < N$ es correcto.

Por lo tanto, se puede concluir que la función es correcta.

Para todo i tal que $0 \leq i \leq N$

(Coloario): el llamado $\text{solve}(0, 0)$ retorna la suma máxima de robots destruidos.

Analisis de la complejidad

- i itera sobre todo el tamaño de x , es decir $0 \leq i \leq N$
- j itera sobre todo el tamaño de f , es decir $0 \leq j \leq N$

Además al usarse memorización nos aseguramos de que no hay recomputaciones de subproblemas
Por lo tanto

El número de subproblemas posibles es:

- i N estados
- j N estados

Como consecuencia el número total de subproblemas es de $N \cdot N$

es decir que la complejidad temporal de la función es de $O(N^2)$