Ejercicio7

Monday, August 26, 2024 2:15 PM

7. Astro Void se dedica a la compra de asteroides. Sea $p \in \mathbb{N}^n$ tal que p_i es el precio de un asteroide el i-ésimo día en una secuencia de n días. Astro Void quiere comprar y vender asteroides durante esos n días de manera tal de obtener la mayor ganancia neta posible. Debido a las dificultades que existen en el transporte y almacenamiento de asteroides, Astro Void puede comprar a lo sumo un asteroide cada día, puede vender a lo sumo un asteroide cada día y comienza sin asteroides. Además, el Ente Regulador Asteroidal impide que Astro Void venda un asteroide que no haya comprado. Queremos encontrar la máxima ganancia neta que puede obtener Astro Void respetando las restricciones indicadas. Por ejemplo, si $p = (3, 2, 5, 6)$ el resultado es 6 y si $p = (3, 6, 10)$ el resultado es 7. Notar que en una solución óptima, Astro Void debe terminar sin asteroides.

b) Escribir matemáticamente la formulación recursiva enunciada en *a*). Dar los valores de los casos base en función de la restricción de que comienza sin asteroides.

$$\text{astroTrade}(p, j, c) = \begin{cases} -\infty & c < 0 \\ -\infty & c > j \\ 0 & j = \text{len}(p) \\ \max\begin{pmatrix} \text{astroTrade}(p, j-1, c-1) + p[j] \\ \text{astroTrade}(p, j-1, c+1) - p[j] \\ \text{astroTrade}(p, j-1, c) \end{pmatrix} & c < \end{cases}$$

```python
def astroTradeBT(P, j, c):
    if c<0:
        return float('-inf')
    if c>j:
        return float('-inf')
    if j==len(P):
        return 0
    return max(astroTradeBT(P, j+1, c-1)+P[j], astroTradeBT(P,j+1,c+1)-P[j],astroTradeBT(P,j+1,c))
```

c) Indicar qué dato es la respuesta al problema con esa formulación recursiva.

astroTrade(len(P), 0) UltimoDia y con 0 asteroides

d) Diseñar un algoritmo de PD *top-down* que resuelva el problema y explicar su complejidad temporal y espacial auxiliar.

```python
def astroTradeTD(P, j, c,mem):
    if c<0:
        return float('-inf')
    if c>j:
        return float('-inf')
    if j==len(P):
        return 0
    if mem[j][c] != None:
        return mem[j][c]
    else:
        mem[j][c] = max(astroTradeTD(P, j+1, c-1,mem)+P[j], astroTradeTD(P,j+1,c+1,mem)-P[j],astroTradeTD(P,j+1,c,mem))
    return mem[j][c]
```

Espacial $O(n^2) \rightarrow$ Matriz $N \times N$

Temporal $O(n^2) \rightarrow O(N \cdot N)$ Llamados distintos

astroTrade($\{0,\dots,N\}$, $\{0,\dots,N\}$)

$O(1)$ Leer de la matriz

e) (Opcional) Diseñar un algoritmo de PD *bottom-up*, reduciendo la complejidad espacial.

```python
def astroTradeBottomUp(P):
    mem = [[float('-inf') for _ in range(len(P)+1)]for _ in range(len(P)+1)]

    mem[0][0] = 0

    for j in range(1, len(P) + 1):
        for c in range(len(P) + 1):
            # No hacer nada
            mem[j][c] = mem[j-1][c]

            # Comprar un asteroide si es posible
            if c > 0:
                mem[j][c] = max(mem[j][c], mem[j-1][c-1] - P[j-1])

            # Vender un asteroide si es posible
            if c < len(P):
                mem[j][c] = max(mem[j][c], mem[j-1][c+1] + P[j-1])

    return mem[len(P)][0]
```

Para reducir complejidad espacial tenemos que darnos cuenta que solo necesitamos la fila anterior

Entonces podemos tener 2 vectores de tamaño N e ir cambiando cual miramos