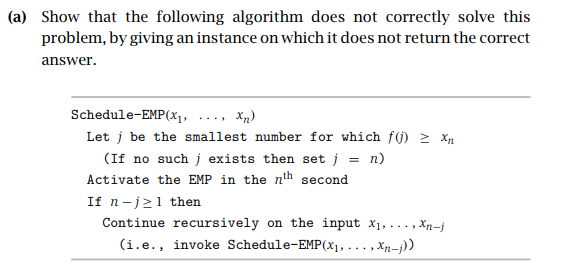
**Tarea 2**

**Problemas conceptuales**

**1. Ejercicio 6.8: The City of Zion (Kleinberg & Tardos, página 319).**



Dado el anterior algoritmo con las siguientes especificaciones:

**Entrada:** Un arreglo [x1, …, xn] con n >= 1

**Salida:** Un valor w entero con w >= 0 devuelve el mayor numero de robot que el rayo EMP puede destruir en t tiempo

Se puede demostrar que no funciona correctamente el algoritmo con el siguiente ejemplo:

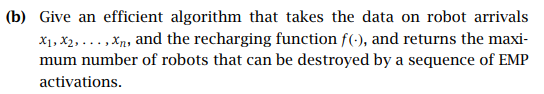
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **i** | **1** | **2** | **3** | **4** | **5** |
| **xi** | **0** | **3** | **0** | **15** | **4** |
| **f(i)** | **1** | **2** | **4** | **8** | **16** |

**Respuesta del algoritmo:**

Lo primero que se realiza en encontrar el j más pequeño tal que f(j) >= Xn , el cual es j = 3 dado que el Xn = 4, posteriormente activa el EMP en el segundo 5, dando como resultado min(4, 16) = 4, luego como la resta n – j = 2, entonces sigue haciendo recursión pero ahora solo con un arreglo que cuenta hasta el segundo 2 y vuelve a hacer la misma comparación, como no hay f(j) >= Xn , entonces j = n, se activa el EMP que da como resultado min(2, 3) = 2, y como n – j = 0, deja de hacer recursión da como resultado **6** robots máximos destruidos.

**Respuesta correcta:**

En el caso anterior y dada la respuesta del algoritmo se nota claramente que la mejor activación del EMP para destruir la mayor cantidad de robots no es en el segundo 2 y 5, realmente debe ser en el segundo 4 pues min(15, 8) = 8 y posteriormente en el segundo 5 pues min(4, 1) = 1, dado como resultado máximo de robots destruidos **9**



Para resolver el problema anterior donde se tenían las siguientes especificaciones:

**Entrada:** Un arreglo [x1, …, xn] con n >= 1

**Salida:** Un valor w entero con w >= 0 devuelve el mayor numero de robot que el rayo EMP puede destruir en t tiempo

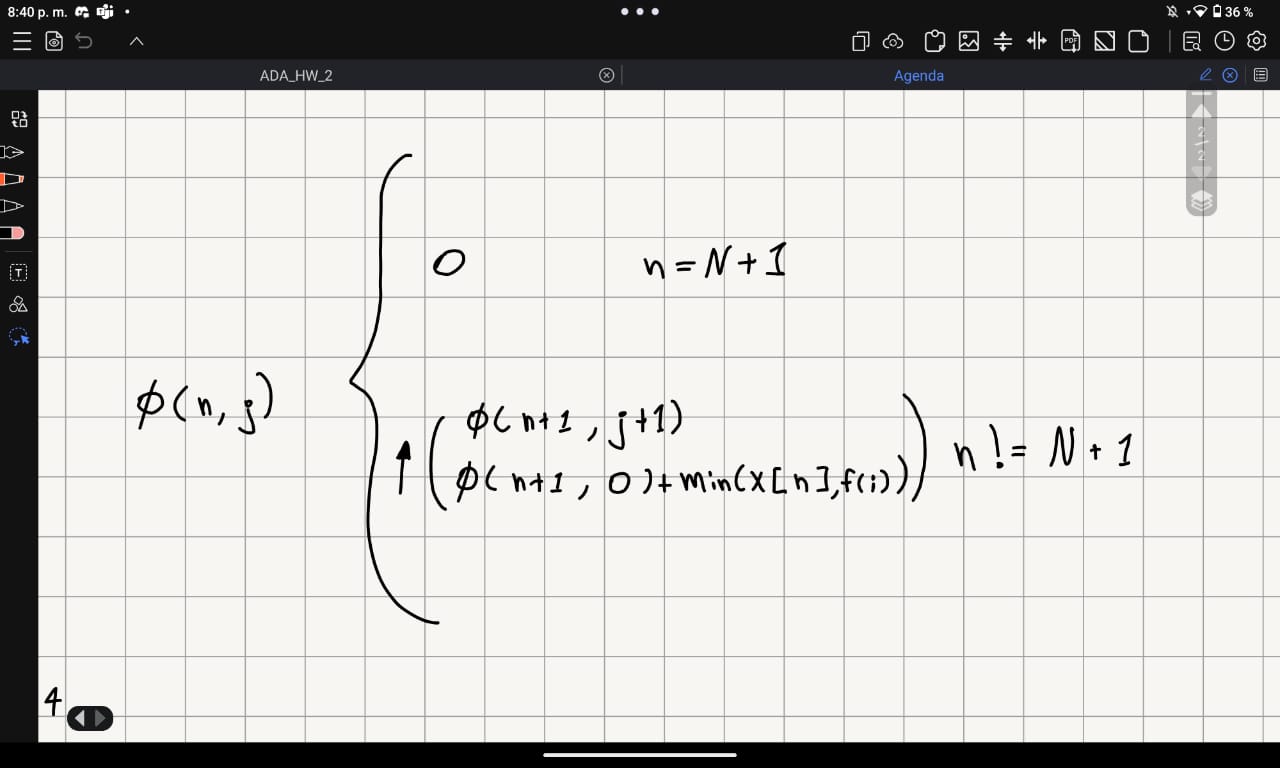
Se propone la siguiente función objetivo: phi(n, j) donde n es el valor índice de los segundos pasados y j el valor que se tiliza en la función f(j)

**Reformulacion:**

**Entrada:** Un arreglo [x1, …, xn] con n >= 1

**Salida:** phi(N, J)

**Planteamiento recursivo:**

****

**Pseudocódigo:**

algoritmo robots(n, j)

si (n , j) esta en memoria entonces:

respuesta = memoria(n,j)

si no:

si n = N + 1 entonces:

respuesta = 0

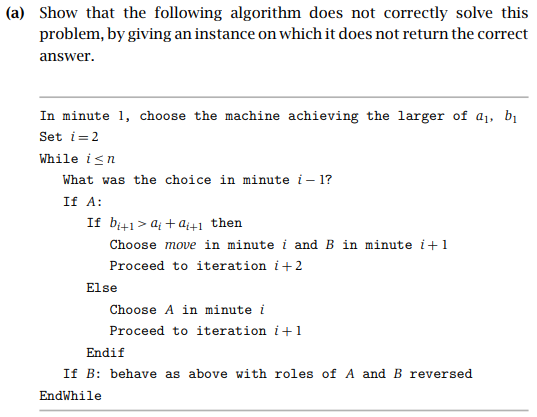
si no:

respuesta = máximo( robots(n + 1, j), robots( n + 1, 0) + min( X[n], f(j) ) )

respuesta = memoria (n , j)

retornar respuesta

**2. Ejercicio 6.10: Large computing jobs (Kleinberg & Tardos, página 321).**

****

Dado el anterior algoritmo con las siguientes especificaciones:

**Entrada:** Un arreglo A [1, …, N] y B [1, …, N] con N >= 1, donde cada posición es el trabajo que realiza una de las 2 máquinas en determinado tiempo

**Salida:** Un valor x entero con x >= 1 devuelve la mayor cantidad de trabajo que se puede realizar entre las 2 maquinas

Se puede demostrar que no funciona correctamente el algoritmo con el siguiente ejemplo:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **N** | **1** | **2** | **3** | **4** |
| **A** | **10** | **3** | **10** | **20** |
| **B** | **5** | **1** | **20** | **5** |

**Respuesta del algoritmo:**

Con el anterior ejemplo lo que el algoritmo primeramente hace es verificar cuál de las dos maquinas A y B hace el mayor trabajo en el minuto 1, en este caso es A llevando una cuenta de +10, posteriormente se para en la posición B[3] y se pregunta si la posición A[2] + A[3] es menor, como si lo es entonces hace un cambio a la maquina B, ahora posicionados en la posición B[3] entonces se lleva una cuenta de +20, y pregunta si A[4] es mayor que B[3] + B[4] como no entonces seguimos con la maquina B llevando al final una cuenta de +10, Al finalizar la ejecución del algoritmo el resultado de las sumas da 35.

**Respuesta correcta:**

En el anterior caso y dada la respuesta del algoritmo se puede notar que claramente no es la respuesta correcta dado que si se quiere obtener la mayor cantidad de trabajo que se realiza en las 2 maquinas realmente se debe usar únicamente la maquina A en todo momento que suma una cantidad de 43 y no 35 como el resultado anterior

****

Para resolver el problema anterior donde se tenían las siguientes especificaciones:

**Entrada:** Un arreglo A [1, …, N] y B [1, …, N] con N >= 1, donde cada posición es el trabajo que realiza una de las 2 máquinas en determinado tiempo

**Salida:** Un valor x entero con x >= 1 devuelve la mayor cantidad de trabajo que se puede realizar entre las 2 maquinas

Se propone la siguiente función objetivo: phi (n, t) donde n es el índice de los arreglos con n >= 1 y t es si estoy en el la maquina A o B, con 1 y 0 respectivamente, devolviendo como resultado el máximo trabajo que se puede realizar.

**Reformulacion:**

**Entrada:** Un arreglo A [1, …, N] y B [1, …, N] con N >= 1, donde cada posición es el trabajo que realiza una de las 2 máquinas en determinado tiempo

**Salida:** phi (N, T)

**Planteamiento recursivo:**

**Imagen que contiene Diagrama

Descripción generada automáticamente**

**Pseudocódigo:**

algoritmo maquinas(n, t)

si (n , t) esta en memoria entonces:

respuesta = memoria(n,t)

si no:

si n = N + 1 o n = N + 2 entonces:

respuesta = 0

si no:

si t = 1 entonces:

respuesta = máximo (maquinas(n+1, t) + A[n], maquinas(n + 2, 0) + A[n])

si no:

respuesta = máximo (maquinas(n+1, t) + B[n], maquinas(n + 2, 1) + B[n])

respuesta = memoria (n , t)

retornar respuesta