Actividad 1.5a

Jorge Emiliano Pomar | A01709338

Tec de Monterrey

Campus Querétaro

TC2038 Algoritmos Avanzados

Profesora:

Ramona Fuentes Valdéz

Fecha de entrega: 27 de febrero de 2024

Problema 1

- Considera la variante de los números de Fibonacci, que denominaremos “números de burronacci”, definida a continuación. El n-ésimo número de burronacci es igual a 4 veces el número (n-1)-ésimo, más 2 veces el (n-2)-ésimo, menos el n-ésimo número de burronacci. El primer y el segundo número de burronacci valen 1 y 2, respectivamente. Se pide lo siguiente:

– Escribe un algoritmo (pseudocódigo) para el cálculo del n-ésimo número de burronacci usando (PD).

```
inicio de la función

     primer numero = 1

     segundo numero = 2

     burronacci [primer numero, segundo numero] = ((primer numero
- 1) * 4) + (2 * (segundo número - 2)) - (primer numero - 3)

     devuelve la función burronacci

fin de la función
```

- Indica claramente la complejidad del algoritmo.

Problema 2

1)

```
para i = infinito
      si i >= cantidad a pagar/cambiar
            se paga con la moneda + (cantidad - moneda)
      si no
            se paga con la moneda + 1
      se regresa el cambio
fin del algoritmo
```

	cantidad a cambiar										
valor/moneda	0	1	2	3	4	5	6	7	8	9	10
1	0	1	2	3	4	5	6	7	8	9	10
4	0	1	1	1	1	2	2	2	2	3	4
9	0	1	1	1	1	1	1	1	1	1	2

2) Para regresar una moneda de diez, se necesitan 2 monedas. Una de 9 y una de 1. Y de esta manera es el menor cambio posible.

Problema 3

1)

	cantidad a cambiar															
valor /moneda	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
10	0	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2
7	0	1	1	1	1	1	1	1	2	2	2	2	2	2	2	3
1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

2) Para entregar el cambio de 15, tendríamos que dar dos monedas de 7 y una moneda de 1. Así das el cambio con la menor cantidad de monedas.

Problema 4

```
Inicio de la función
     Recibe parámetro n
      si n es igual a 0
            devuelve 1
      si no es 0
            P(n - 1)

     devuelve n * P(n - 1)
fin de la función
```

Este algoritmo usa el sistema recurrente y la idea es que por cada iteración se vayan guardando las iteraciones pasadas y de esta manera hacerlo dinámico. Y por esto creo que la complejidad sería O(n). Porque en el peor de los casos depende directamente de lo que se ingrese a la función.