Taller I

Fundamentos de Análisis y Diseño de Algoritmos / 750094M /
Profesor: Jesús Alexander Aranda Bueno /
Monitor: Reynell Arkad Devji Quevedo Pereira/ 2021-2

I. Análisis de Algoritmos (2 puntos)

Punto 1. En cada literal se encontrará un algoritmo escrito en pseudocódigo. Para cada algoritmo, debe expresar su orden de complejidad en notación *Theta*, es decir en términos de Θ(___). La operación básica a considerar en los siguientes algoritmos es **print("Hola Mundo")**:

(a)
```
funcion1(n)
        for(i = 2; i <= 5; i = i * i)
                print("Hola Mundo")
        for(i = 1; i <= n; i = i * 5 )
                print("Hola Mundo")
```

(b)
```
funcion2(n)
        for(i = 1; i <= n; i = i * 2)
                for(j = 1; j <= n; j = j+j)
                        print("Hola  Mundo")
```

(c)
```
funcion3(n)
        if(n == 1) then
        { print("Hola Mundo") }
        else
        { print("Hola Mundo")
          funcion3(n/2)
```

```
            funcion3(n/2)}
```

(d)
```
funcion4(n)
          if(n == 1) then
          { print("Hola Mundo") }
          else
          { print("Hola Mundo")
            for(j = 1; j <= 3; j = j + 1)
                   funcion4(n/3)}
```

Punto 2. Dado el siguiente algoritmo:

```
1. FuncionMisterio(a, b)
2. if b == 0
3.    return 1
4. if b == 1
5.    return a
6. if even(b)
7.    c = FuncionMisterio(a, b/2)
8.    return c*c
9. else
10.   c = FuncionMisterio(a, b/2)
11.   return c*c*a
```

Donde a y b son números enteros positivos

1. Explique qué hace el algoritmo (asuma que existe una función even que recibe un número y devuelve true si es par, false si es impar, y lo hace en O(1)). Asuma que a y b son 2 valores enteros positivos.
2. ¿Cuál es el mínimo de veces que se ejecuta la línea 7? ¿Cuál es el máximo? ¿De qué depende las veces que se ejecute en las siguientes llamadas al método?
3. Diga la complejidad del algoritmo.

Punto 3. Dado el siguiente algoritmo:

```
1. FuncionMisterio(n, arr[1..n])
2.   for i = 2 to n, i = i + 1
3.        if(arr[i] ¡= true)
4.             for j = 2*i to n, j = j + i
5.                  arr[j] = true
```

Donde n es un entero y arr un arreglo de booleanos de tamaño n, inicializado en false
Escriba y explique la complejidad del algoritmo presentado.

II. Diseño de Algoritmos (3 puntos)

Punto 4. Creación de equipos

Usted tiene a su disposición n estudiantes que desea agrupar en pares para un torneo de un juego. El torneo consta de 2 juegos llamados A y B, y cuenta con los datos de cada estudiante de "su nivel" en cada juego en el par (a, b), siendo su nivel en el juego A y B respectivamente. Lógicamente existen mejores jugadores en un juego que en otro, y jugadores que tienen capacidades inferiores que otros en ambos juegos, por lo que definiremos el nivel de una pareja como la suma de sus niveles en los juegos (Ejemplo, si tenemos un estudiante 1 con características (a1, b1) y otro con capacidad (a2, b2), la capacidad de la pareja es (a1+a2, b1+b2)). Como quiere hacer el torneo lo más justo posible, le gustaría saber si dados n estudiantes puede agruparlos en pares de tal forma que todas las parejas tengan el mismo nivel. Retorne true o false en caso afirmativo o negativo respectivamente

Formato de entrada:

- N -> Número de estudiantes, entero
- E: arreglo de objetos con valor a y b, representado como un par

Ejemplos:

Entrada:

- N : 6
- E: [(2,1), (4,2), (3,0), (4,2), (3,0), (5,1)]

Salida: true

Se pueden hacer en parejas los equipos (2,1) y (5,1), (4,2) y (3,0), (4,2) y (3,0), quedando todos con nivel (7,2)

Entrada:

- N: 4
- E: [(1, 0), (0, 1), (-2, 0), (0, -2)]
-

Salida: false

No se pueden formar en parejas de igual nivel

Se espera que el algoritmo tenga una complejidad de O(nlogn), para lo cual se necesita:

1. Precisar de manera correcta la idea de su algoritmo.
2. Realizar un pseudo-código del algoritmo.
3. Estimar la complejidad del algoritmo en base al pseudo-código.

Punto 5. Gestión de apartamentos

Usted tiene m apartamentos libres, con precios M_i (**0 < i ≤ m**) de arrendamiento cada uno, y tiene n personas que pidieron arrendar alguno de los m apartamentos, y cada aplicante le dio un precio N_j (**0 < j ≤ n**) que está dispuesto a pagar. Sin embargo, este precio tiene una tolerancia k para todos lo que implica que están dispuestos a pagar desde N_j – k hasta N_j + k. Determine la mayor cantidad de apartamentos que puede arrendar. **Asuma que m ≤ n**.

Entradas:

- m: 3 -> Número de apartamentos
- n: 4 -> Número de aplicantes
- k: 5 -> Tolerancia en el precio de los aplicantes
- M: [30, 60, 75] -> Precios de los apartamentos
- N: [60, 45, 80, 60] -> Dinero tentativo de los aplicantes

Salida: 2

El primer aplicante y el tercero pueden alquilar un apartamento, uno paga 60 declarando que puede pagar 60 y el otro paga 75 declarando que puede pagar 75

Se espera que el algoritmo tenga una complejidad de O(nlogn), para lo cual se necesita:

1. Precisar de manera correcta la idea de su algoritmo.
2. Realizar un pseudo-código del algoritmo.
3. Estimar la complejidad del algoritmo en base al pseudo-código.

Punto 6. Tanque de gasolina vacío

Usted viaja con un tanque de gasolina N completamente lleno, donde cada día gastas gasolina igual al número de días que han transcurrido (el día 1 gastas 1 de gasolina, el día 2 gastas 2 de gasolina, el día i-ésimo gastas i de gasolina). Al inicio de cada día recargarás R de gasolina o hasta que se llene el tanque, **lo que ocurra primero**. ¿Cuál es el primer día en el que te quedarás sin gasolina? Imprima el número de ese día

Ejemplo:

Entrada:

- N: 5
- R = 2

Salida: 4

El primer día se acaba con (5 - 1) = 4 de gasolina.

Al inicio del segundo día recargas 2, sin embargo la capacidad máxima es 5, por lo que empiezas con min(4 + 2, 5) = 5. Terminas el día con 5-2 = 3 de gasolina.

En el tercer día recargas y tienes 5 de gasolina, acabas el día con 5-3 = 2 de gasolina.

En el cuarto día recargas y empiezas con 4 de gasolina, acabas el día con 4 – 4 = 0 de gasolina, por lo que la respuesta es 4.

Se espera que el algoritmo tenga una complejidad de O(logn), para lo cual se necesita:

1. Precisar de manera correcta la idea de su algoritmo.
2. Realizar un pseudo-código del algoritmo.
3. Estimar la complejidad del algoritmo en base al pseudo-código.

Aclaraciones sobre el taller

El taller se puede desarrollar en grupos de máximo 4 estudiantes; el taller necesita ser entregado por medio del campus virtual a más tardar el 13 de enero. El documento correspondiente al taller debe indicar el nombre de cada uno de los estudiantes que desarrollaron el taller. Todas las soluciones del taller necesitan su respectiva explicación, excepto el punto 1.