Estructuras de Datos y Algoritmos 1 - ST0245
Primer Parcial (001, 002 y 003)

Nombre .....
Departamento de Informática y Sistemas
Universidad EAFIT

Septiembre 22 de 2020

1 Mujeres en Ingeniería de Sistemas (2% extra)

(2 %) Varios estudios argumentan que muchas mujeres deciden NO estudiar ingeniería de sistemas porque creen en los estereotipos sobre el tipo de personas que trabajan en el campo, y no se ven a sí mismas encajando en esos estereotipos. De esta manera, incorrectas percepciones pueden dar forma a las trayectorias profesionales. Una forma de desmentir dichos estereotipos es reconociendo mujeres exitosas en ingeniería de sistemas –tanto a nivel nacional como mundial. Para lograr esto, relaciona las siguientes mujeres con su contribución:

1. Marrisa Meyer	a. Primera de Eafit en pasar a Google
2. María C. Choucair	b. Primera gerente general de Yahoo
3. Ana Echavarría	c. Primera de Eafit en pasar a Facebook
4. Kathie Bouman	d. Código para el Apolo 11
5. Margaret Hamilton	e. Primera foto de un agujero negro
6. Luisa M. Vásquez	f. Dueña de la empresa de *testing* más grande de Colombia

1..., 2..., 3..., 4..., 5..., 6...

2 Complejidad (2% extra)

(2 %) En la vida real, los memes se utilizan en presentaciones orales como una estrategia para *romper el hielo*. Con base en los errores –sobre complejidad– que has cometido durante el semestre, escribe un texto para el siguiente meme. Como un ejemplo, ten en cuenta la regla de la suma, regla del producto, notación O o ecuaciones de recurrencia.

....
....
....
....
....
....
....

3 Recursión 30%

Rellenar por difusión se utiliza para implementar el tarrito de *Microsoft Paint*. Además, según el portal *Geeks for Geeks*, es un problema –muy frecuente– en entrevistas de Amazon, Microsoft y Adobe. El problema es el siguiente. Dado una matriz de dimensión $N \times M$ (`screen`), la ubicación de un pixel en la matriz (x, y) y un color (`newC` $\in \mathbb{N}$), nuestra tarea es reemplazar el color (`prevC` $\in \mathbb{N}$) del pixel (x, y) –y el de todos sus vecinos que tengan el color `prevC`– con el color `newC`. **¡Incluyamos los vecinos diagonales!** Como un ejemplo, si tenemos $x = 4, y = 4, prevC = 2, newC = 3$ y la matriz (`screen`) en el lado izquierdo, después de aplicar el algoritmo, debe lucir como la del lado derecho. Faltan unas líneas, complétalas, por favor.

```
{{1, 1, 1, 1, 1, 1, 1, 1},   {{1, 1, 1, 1, 1, 1, 1, 1},
{1, 1, 1, 1, 1, 1, 0, 0},    {1, 1, 1, 1, 1, 1, 0, 0},
{1, 0, 0, 1, 1, 0, 1, 1},    {1, 0, 0, 1, 1, 0, 1, 1},
{1, 2, 2, 2, 2, 0, 1, 0},    {1, 3, 3, 3, 3, 0, 1, 0},
{1, 1, 1, 2, 2, 0, 1, 0},    {1, 1, 1, 3, 3, 0, 1, 0},
{1, 1, 1, 2, 2, 2, 2, 0},    {1, 1, 1, 3, 3, 3, 3, 0},
{1, 1, 1, 1, 1, 2, 1, 1},    {1, 1, 1, 1, 1, 3, 1, 1},
{1, 1, 1, 1, 1, 2, 2, 1},}; {1, 1, 1, 1, 1, 3, 3, 1},};
```

Si trabajas en Java, revisa este código:

```
1  void floodFillUtil(int screen[][], int x, int
       y, int prevC, int newC, int N, int M) {
2    if (x < 0 || x >= M || y < 0 || y >= N)
3          return;
4    if (screen[x][y] != prevC)
5          return;
6
7    screen[x][y] = newC;
8
9    ....
10   ....
11   ....
12   ....
13   floodFillUtil(screen, x+1, y, prevC, newC, N,
         M);
14   floodFillUtil(screen, x-1, y, prevC, newC, N,
         M);
15   floodFillUtil(screen, x, y+1, prevC, newC, N,
         M);
16   floodFillUtil(screen, x, y-1, prevC, newC, N,
         M); }
17
18 void floodFill(int screen[][], int x, int y,
       int newC, int N, int M) {
19   int prevC = screen[x][y];
20   floodFillUtil(screen, x, y, prevC, newC, N, M
         ); }
```

Si trabajas en Python, revisa este código:

```
1  def floodFillUtil(screen, x, y, prevC, newC, N
       , M):
2    if x < 0 or x >= M or y < 0 or y >= N or
        screen[x][y] != prevC:
3          return
4    if screen[x][y] == newC:
5          return
6
7    screen[x][y] = newC
8
9    ....
10   ....
11   ....
12   ....
13   floodFillUtil(screen, x + 1, y, prevC, newC,
       N, M)
14   floodFillUtil(screen, x - 1, y, prevC, newC,
       N, M)
15   floodFillUtil(screen, x, y + 1, prevC, newC,
       N, M)
16   floodFillUtil(screen, x, y - 1, prevC, newC,
       N, M)
17
18 def floodFill(screen, x, y, newC, N, M):
19   prevC = screen[x][y]
20   floodFillUtil(screen, x, y, prevC, newC, N, M
       )
```

A (10%) Completa la línea 9 .....
Y Completa la línea 10 .....

B (10%) Completa la línea 11 .....
Y Completa la línea 12 .....

C (10%) ¿Cuál es la ecuación de recurrencia que representa la complejidad, en el tiempo, para el peor de los casos, en términos de $p = n + m$?

T(p) = ...

4 Notación O 20%

La *lógistica de última milla* es el proceso de entregar un pedido de comercio electrónico (por ejemplo, en Amazon o Rappi) desde que sale de la tienda hasta que llega al cliente final. La logística de última milla representa hasta un 50% de los costos logísticos. Contar el número de caminos se utiliza en logística de última milla. Además, según el portal *Geeks for Geeks*, este es un problema –muy frecuente– en entrevistas de Amazon, Microsoft y y Adobe. El problema es contar todos los posibles caminos, desde la esquina superior izquierda $(0, 0)$ hasta la esquina inferior derecha $(n-1, m-1)$, de una matriz de $n \times m$, con la restricción de que desde cada celda sólo nos podemos mover hacia la derecha $(i, j+1)$ o hacia abajo $(i+1, j)$. Este problema es similar al problema del conejo, ¿cierto? Como un ejemplo, si $n = 3$ y $m = 2$, la salida es 3 porque hay estos 3 caminos:

- $(0,0) -> (0,1) -> (0,2) -> (1,2)$
- $(0,0) -> (0,1) -> (1,1) -> (1,2)$
- $(0,0) -> (1,0) -> (1,1) -> (1,2)$

Si trabajas en Java, considera este código:

```
1  int numberOfPaths(int m, int n) {
2    int count[][] = new int[m][n];
3
4    for (int i = 0; i < m; i++)
5          count[i][0] = 1;
6
7    for (int j = 0; j < n; j++)
8          count[0][j] = 1;
9
10   for (int i = 1; i < m; i++) {
11     for (int j = 1; j < n; j++)
12       count[i][j]=count[i-1][j]+count[i][j-1];
13   }
14   return count[m - 1][n - 1];
15 }
```

Si trabajas en Python, considera este código:

```
1  def numberOfPaths(m, n):
2    count = [[0 for y in range(n)] for x in range
         (m)]
3
4    for i in range(m):
5      count[i][0] = 1
6
7    for j in range(n):
8      count[0][j] = 1
9
10   for i in range(1, m):
11     for j in range(1, n):
12       count[i][j] = count[i-1][j] + count[i][j-1]
13   return count[m-1][n-1]
```

A (10%) ¿Cuál es la complejidad asintótica, en **tiempo**, en el peor de los casos, en términos de n y de m?
O(.....)

B (10%) ¿Cuál es la complejidad asintótica, en **memoria**, en el peor de los casos, en términos de n y de m?
O(.....)

La complejidad en memoria quiere decir cuántos enteros nuevos crea el algoritmo, fuera de los que ya existían.

5 Listas enlazadas 20%

Vamos a encontrar la mediana de dos listas ordenadas. Según el portal *Leet Code*, este un problema –muy frecuente– en entrevistas de Google, Facebook y Apple. Dadas dos listas ordenadas, ar_1 de tamaño N y ar_2 de tamaño M, la tarea es encontrar la mediana de estas dos listas juntas. La mediana se define como el valor que queda en la mitad cuando se juntan ambas listas en una nueva lista ordenada. Si el número de elementos es par, la mediana se calcula sumando los dos valores de la mitad y dividiéndolo en dos. La mediana tiene amplias aplicaciones en Estadística y *Machine Learning*[1], pues se ha encontrado que es un estimador robusto de la media poblacional; es decir, que no se afecta tanto por datos atítpicos como la media.

Como un ejemplo, si la primera lista enlazada ar_1 es $15- > 10- > 5$ y la otra lista enlazada ar_2 es $20- > 3- > 2$, entonces ambas listas unidas son $20- > 15- > 10- > 5- > 3- > 2$ y la mediana es $(5 + 10)/2 = 7.5$.

Si trabajas en Java, considera este código:

```
1  int i, j = 0;
2  int aux(LinkedList<Integer> ar1, LinkedList<
       Integer> ar2, int n, int m){
3   int m1 = 0;
4   if(i != n && j != m)
5    ....
6   else if(i < n)
7    m1 = ar1.get(i++);
8   else
9    m1 = ar2.get(j++);
10  return m1;
11 }
12
13 float med(LinkedList<Integer> ar1, LinkedList<
       Integer> ar2){
14  int count;
15  i = 0; j = 0;
16  int m1 = -1, m2 = -1, n = ar1.size(), m = ar2
       .size();
17
18  if((m + n) % 2 == 1)  {
19   for (count = 0; count <= (n + m)/2; count++)
20     m1 = aux(ar1, ar2, n, m);
21   return m1;
22  }
23  else {
24   for(count = 0; count <= (n + m)/2; count++){
25     m2 = m1;
26     m1 = aux(ar1, ar2, n, m);
27   }
28   return (m1 + m2)/2f;
29  }
30 }
```

En Java, la asignación `variable = pregunta ? verdadero : falso;` funciona de la siguiente forma: si la pregunta es verdadera, asigna verdadero a la variable, de lo contrario asigna falso. Como un ejemplo `maximo = a > b ? a : b;`

Si trabajas en Python, considera este código:

```
1  class Node:
2      def __init__(self, data):
3          self.data = data
4          self.next = None
5
6  class LinkedList:
7      def __init__(self, h):
8          self.head = h
9      def get(self, i):
10         temp = self.head
11         for i in range(0,i):
12           temp = temp.next
13         return temp.data
14
15 def aux(ar1, ar2,  n,  m, i, j):
16  m1 = 0
17  if i != n and j != m:
18    if ....:
19          ....
20          ....
21    else:
22          ....
23          ....
24  elif i < n:
25   m1 = ar1.get(i)
26   i = i+1
27  else:
28   m1 = ar2.get(j)
29   j = j+1
30  return m1, i, j;
31
32 def med(ar1,  ar2, n, m):
33  i, j, m1, m2 = 0, 0, -1, -1
34  if (m + n) % 2 == 1:
35   for count in range(0, int((n+m)/2+1)):
36     m1,i,j = aux(ar1, ar2, n, m, i, j)
37   return m1
38  else:
39   for count in range(0, int((n+m)/2)+1):
40     m2 = m1
41     m1,i,j = aux(ar1, ar2, n, m, i, j)
42  return (m1 + m2)/2
```

A (10%) ¿Cuál es la complejidad asintótica, en el peor de los casos, del algoritmo anterior, en términos de $p = n + m$? O(.....)

B (10%) Completa la línea 5 en Java ..... o las líneas 18, 19, 20, 22 y 23 en Python .....;.....;.....;.....;.....

El método `a.get(i)` retorna el elemento en la posición i de una lista enlazada a. ¿Recuerdas qué complejidad tiene el método `get` en listas enlazadas?

6 Vectores dinámicos 10%

Cuando enviamos información a través de internet, por ejemplo, del videojuego *Valorant* al servidor servidor de *Riot Games*, lo normal es enviar la información en forma de cadenas de caracteres. Una vez llega al servidor, tenemos que convertir esa información en números. Convertir una cadena de caracteres en un número es un problema –frecuente– en entrevistas de Snapchat, Oracle y Uber, según el portal *Leet Code*. Para este ejercicio, implementa un algoritmo que permita convertir un vector dinámico

[1] Velasco, H.; Laniado, H.; Toro, M.; Leiva, V.; Lio, Y. Robust Three-Step Regression Based on Comedian and Its Performance in Cell-Wise and Case-Wise Outliers. Mathematics 2020, 8, 1259.

de caracteres (que contiene un número binario) en un entero decimal. Supongamos que no hay espacios en blanco, el número es positivo, el número es entero y que el vector sólo contiene los caracteres '0' y '1'. Como un ejemplo, para el vector $[1, 0, 1, 1]$ su equivalente en decimal es $1 \times 2^0 + 1 \times 2^1 + 0 \times 2^2 + 1 \times 2^3 = 11$.

Si trabajas en Java, considera este código:

```
1  int convertir(ArrayList<Character> str){
2   int res = 0;
3   for (int i = 0; i < str.size(); ++i)
4     ....
5   return res;
6  }
```

En Java, el método `a.get(i)` retorna el elemento en la posición i del vector a. Para convertir un caracter numérico c en su equivalente en entero se hace la resta c - '0', donde c es el caracter que queremos convertir y '0' es el caracter que representa el entero 0.

Si trabajas en Python, considera este código:

```
1  def convertir(vector):
2      res = 0
3      for i in range(len(vector)):
4          ....
5      return res
```

En Python, la función `int(c)` convierte el caracter c en su valor entero; por ejemplo, convierte '2' en 2. Y el operador `a[i]` retorna el elemento en la posición i del vector dinámico a.

A (10%) Completa la línea 4 .....
Y ¿Cuál es la complejidad asintótica, en el peor de los casos, del algoritmo anterior? O(.....)

7 Complejidad 20%

Las listas enlazadas se utilizan para implementar colas. Las colas se utilizan en simulación por colas. La simulación por colas la podrías utilizar para modelar la llegada de los carros a Eafit, y cómo una tarifa diferenciada –dependiendo la hora de llegada y el número de pasajeros en vehículo– afectaría el tiempo promedio de espera. Para este ejercicio, vamos a juntar dos listas enlazadas ordenadas (A y B), de menor a mayor, en una nueva lista enlazada que también será de menor a mayor. La nueva lista enlazada se construye juntando los nodos de las dos listas originales. Este problema es común en entrevistas de Riot Games, Snapchat y Dropbox, según el portal *Geeks for Geeks*. Como un ejemplo, si la primera lista enlazada A es $5- > 10- > 15$ y la otra lista enlazada B es $2- > 3- > 20$, entonces `sortedMerge(A,B)` debe retornar una nueva lista $2- > 3- > 5- > 10- > 15- > 20$.

Si trabajas en Java, considera este código:

```
1  class Node{
2    public int data;
3    public Node next;
4    public Node(int d){
5      data = d; next = null;
6    }
7  }
8
9  Node sortedMerge(Node A, Node B) {
10   if(A == null) return B;
11   if(B == null) return A;
12   if(A.data < B.data) {
13     A.next = sortedMerge(A.next, B);
14     return A;
15   }
16   else {
17     B.next = sortedMerge(A, B.next);
18     return B;
19   }
20 }
```

Si trabajas en Python, considera este código:

```
1  class Node:
2      def __init__(self, data):
3          self.data = data
4          self.next = None
5
6  def sortedMerge(A, B):
7    if A == None:
8     return B
9    if B == None:
10    return A
11   if A.data < B.data:
12     A.next = sortedMerge(A.next, B)
13     return A
14   else:
15     B.next = sortedMerge(A, B.next);
16     return B;
```

A (10%) ¿Cuál es la complejidad, en **tiempo**, en el peor de los casos, en términos de $p = n + m$? T(p) = ...

B (10%) Si sabemos que $T(p) = c + T(p-1)$ es $O(p)$, $T(p) = c + 2T(p-1)$ es $O(2^p)$, $T(p) = c + T(p/2)$ es $O(\log_2 p)$ y $T(p) = c$ es $O(1)$, ¿cuál es la complejidad asintótica, en **tiempo**, en el peor de los casos, en términos de $p = n + m$? O(.....)