ESTRUCTURA DE DATOS 1
Código ST0245

Laboratorio Nro. 1
Recursión

Miguel Angel Martinez Florez
Universidad Eafit
Medellín, Colombia
mamartinef@eafit.edu.co

Pablo Maya Villegas
Universidad Eafit
Medellín, Colombia
pmayav@eafit.edu.co

3) Simulacro de preguntas de sustentación de Proyectos

3.1. Calculen la complejidad asintótica, para el peor de los casos, del ejercicio 1.1

R El algoritmo recursivo tiene una razón de cambió de 2^X, esto significa que entre más grande la cantidad que se evalué, se va a demorar más y los números altos se pueden demorar hasta años en terminarse de procesar, un solo aumento de X incrementa potencialmente el tiempo y ciclos que realiza.

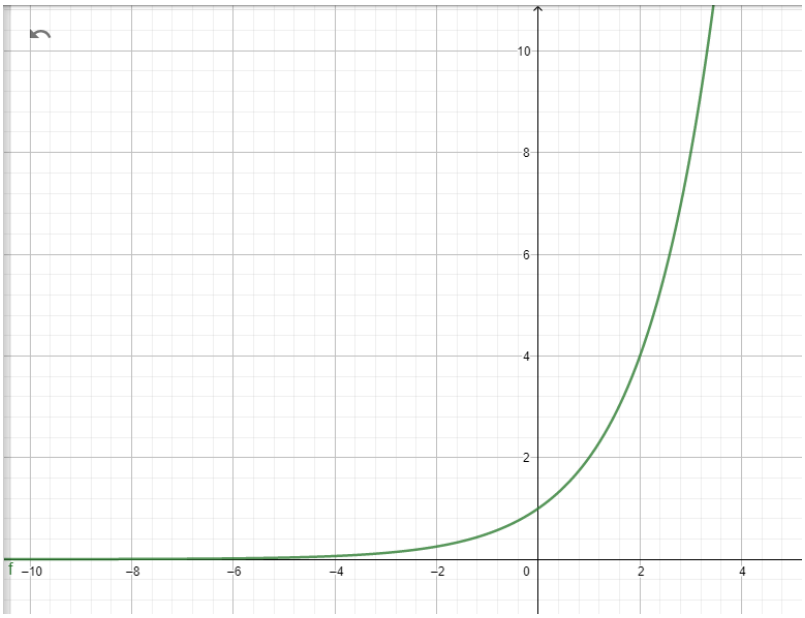
3.2. Para el ejercicio 1.1, tomen tiempos para 20 tamaños del problema diferentes, generen una gráfica y analicen los resultados. Estimen cuánto tiempo se demorará este algoritmo en la subsecuencia común más larga entre dos ADNs mitocondriales (que tienen alrededor de 300.000 caracteres cada uno)

R

Tamaño:	Recursiones:	Tiempo:
11	26667	20
12	106006	34
13	318032	46
14	582732	67
15	794759	62
16	3438617	137
17	12432260	363
18	35495254	1037
19	85327171	1920
20	277044648	6717
21	568425981	12895
22	760143459	12787

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

23	2872908681	63002
24	3064626161	59929
25	5177391384	61909
26	9494975486	177372
27	11607740710	173931
28	15925324813	171988
29	44919490555	887665
30	1,60823E+11	3879457



Esta es la gráfica de 2^X confirmando la semejanza de relación.

3.3. ¿La complejidad del algoritmo del ejercicio 1.1? es apropiada para encontrar la subsecuencia común más larga entre ADNs mitocondriales como los de los datasets?

R Este algoritmo no es adecuado para calcular 300mil valores, debido a que es potencial, ese valor se demora un tiempo infinito, otro algoritmo, aunque sea más lento en corto plazo puede tener un mayor potencial para calcular esa magnitud

3.4. [Ejercicio opcional] Expliquen con sus propias palabras cómo funciona GroupSum5

R. Para este punto se deben de tomar todos los múltiplos de 5 en la suma, y si un 1 les sigue directamente este no se puede contar.
Para lograr esto vamos a procesar todos los números del array, y si uno es múltiplo de 5 entonces se le resta su valor a la meta final y se transforma en 0. Justo después de que se

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

descubrió un múltiplo de 5 se observa si el siguiente número del array es un 1, tener en cuenta que si estamos en el último valor no se hace este paso y si se confirma que el siguiente es un 1, entonces lo convertimos en un 0.
Cuando ya todo el array este ordenado y la meta reducida vamos a verificar que esta no sea negativa, en caso contrario es imposible la suma, y ya solo se llama al método recursivo groupSum para que compruebe que con el resto de los números sea posible alcanzar la meta.

3.5. Calculen la complejidad de los Ejercicios en Línea de los numerales 2.1 y 2.2
R

- groupNoAdj = $T(5n+3n^2)$
- fibonacci = $T(2^n)$
- countX = $T(3+n)$
- groupSumClump = $T(2+12n+3n^2)$
- array11 = $T(3+n)$
- splitOdd10 = $T(n+2^n)$
- noX = $T(3+n)$
- split53 = $T(3n+2^n)$
- changeYX = $T(3+n)$
- groupSum5 = $T(1+6n+2^n)$

3.6. Calculen la complejidad de los Ejercicios en Línea de los numerales 2.1 y 2.2

R\ La n es la magnitud de la entrada al método, si es un string de 8 caracteres, entonces n es 8; y m es las veces que se hace un incremento, por así decirlo, un limitante de cuantos ciclos son permitidos.

4) Simulacro de Parcial

4.1 Una de las tecnologías cruciales para que la Cuarta Revolución Industrial tenga éxito es laciberseguridad.Una de las áreas másimportantes de la ciberseguridad es lacriptografía. Criptografía es el arte y técnica de escribir claves secretas de tal forma que loescrito solamente sea inteligible para quien sepa descifrarlo. Consideremos una aplicación de criptografía. Algunas palabras del alfabeto español se pueden escribir como la unión consecutivade los símbolos de la tabla periódica universal. Por ejemplo, la palabra “Población” se puede escribir como la unión de los símbolos {Po, B, La, C, I, O, N}, PoBLaCION. Te entregan una cadena de caracteresSy un conjunto de símbolosT. El objetivo es determinar siSse puede escribir como la unión consecutiva de cero o más símbolos deT. Por ejemplo, seaT={Ti,B,I,O,C} yS= “Biótico”, larespuesta es verdadero; paraS= “”, la respuesta es verdadero; y paraS=“Titan”, la respuesta es falso. El siguiente código resuelve el problema, pero le faltan algunas líneas; por favor, complétalas. ¡Gracias! Puedes asumir que T contiene todas las posibles combinaciones de acentos, mayúsculas y minúsculas de cada símbolo de la tabla periódica. En Java, el

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

método s.substring(a,b) retorna la subcadena de s entre los índices a y b−1, incluidos. El método t.contains(s) retorna verdadero si la cadena s se encuentra dentro de t; de lo contrario, falso.


```
1   boolean solve(String s,List<String> t){
2      return solve(t, s, s.length());
3   }
4   boolean solve(List<String> t, String s,
        int n){
5       for(int i = 1; i <= n; ++i){
6         String pfx = ..................;
7         if(t.contains(pfx)){
8           if(i == n){
9             return .............;
10          }
11          return ......................;
12        }
13      }
14     return false;
15  }
```

1. Completa la línea 6 .....
a. s.substring(0, i)
b. s.substring(0, n)
c. s.substring(i, n)

R\ c. s.substring(i, n)

2. Completa la línea 9 .....
a. false
b. s.substring(0, i)
c. true

R\ c. true

3. Completa la línea 11 .....
a. solve(t, s.substring(i), n - i)
b. solve(pfx, t), n - i)
c. solve(t, s.substring(n), I - n)

R\ a. solve(t, s.substring(i), n - i)

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

4.2 En la vida real, empresas como Riot Games hacen un uso extensivo de bases de datos. En muchas bases de datos, los tiempos de búsqueda son O(log n).

(10%) Sea c una constante, $f(n)$ es $O(g(n))$ si $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$, donde $0 < c < \infty$.

.

1.
(a)Verdadero
(b)Falso

R\ a) Verdadero

2. (10%) Determina cuales de las siguientes proposiciones son verdaderas. Varias pueden ser verdaderas.
(a)Si f(n) es O(g(n)) y g(n) es O(h(n)), entonces f(n) es O(h(n)).
(b)Para cualquier base b>0, es O(ln(n)).
(c)Para cualquier constante c, c es O(c).
(d)Sean y . Entonces,

R\ (c)Para cualquier constante c, c es O(c).

¡Recuerda la regla de cambio de base de los logaritmos!

4.3 En la vida real, las grandes empresas de tecnología, solicitan calcular la complejidad de los algoritmos que preguntan en sus entrevistas técnicas; por ejemplo, Google, Facebook y Riot Games. Estas entrevistas se realizan para acceder a una práctica laboral como lo hizo Santiago Zubieta, en 2015, y Juan M. Ciro, en 2019, para hacer sus prácticas en Google y Facebook, respectivamente. ¡Ambos eran estudiantes de la Universidad Eafit! Para prepararnos para esas entrevistas, para cada uno de los siguientes códigos, determina la ecuación que representa su complejidad para el peor caso. Para el siguiente código, determina la ecuación que representa su complejidad para el peor caso. Considera que C es una constante.

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

```
1    void mistery(int n, int m){
2      int res = 0;
3      for(int i = 0; i < n; ++i){
4        for(int j = 1; j < m; ++j){
5          for(int k = 1; k < m; ++k){
6            res = res + 1;
7    } } } }
```

La complejidad de la función mistery es
a. T(n,m) = C x n x m
b. T(n,m) = C x n x m^2
c. T(n,m) = C x n x mlogm
d. T(n,m) = C x n x m^3
R\ b. T(n,m) = C x n x m^2

4.4 Lika y Kefo están estudiando para el examen de matemáticas en su colegio. Hoy, ellos encontraron muchas secuencias de números interesantes, una de ellas los números Fibonacci. Para Lika –que ya conocía estos números– se le hace aburrido escribirlos todos; sin embargo, ellos encontraron otra secuencia de números llamados los números de Lucas. En el libro donde ellos encontraron esta secuencia de números, sólo hay 7 números pertenecientes a la secuencia y al final de la hoja dice: "¿Podrás definir una relación de recurrencia que nos permita deducir el n-ésimo número de Lucas?" "Por supuesto" –dijeron ambos. Ahora, ellos quieren escribir un algoritmo que nos permita retornar el n-ésimo número de Lucas. ¿Puedes ayudarlos a escribir el algoritmo? La secuencia encontrada fue la siguiente: 2, 1, 3, 4, 7, 11, 18, …. Se sabe que el número 2 y el número 1 son el primer y segundo término, respectivamente, de la secuencia de los números de Lucas. Al código le faltan algunas líneas, para completar el ejercicio deberás completarlas.

```
1 int lucas(int n){
2   if(n == 0) return 2;
3   if(n == 1) return 1;
4   return lucas(______) + ______;
5 }
```

4.4.1 La complejidad asintótica del algoritmo anterior, para el peor de los casos, en términos de n, es:
a. T(n)=T(n−1)+c, que es O(n)
b. T(n)=4T(n/2)+c, que es O(𝑛2)

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

c. T(n)=T(n−1)+T(n−2)+c, que es O(2^n)
d. T(n)=c, que es O(1)
R\ C. T(n)=T(n−1)+T(n−2)+c, que es O(2^n)

4.5 *En la vida real, los palíndromes se utilizan para desarrollar algoritmos de compresión de cadenas de ADN. Para este parcial, considera un algoritmo capaz de decir si una cadena de caracteres es un palíndrome o no. Un palíndrome es una cadena que se lee igual de izquierda a derecha que de derecha a izquierda. A continuación, algunos ejemplos:*
• Para "amor a roma", la respuesta es true
• Para "mamita", la respuesta es false
• Para "cocoococ", la respuesta es true

El algoritmo isPal soluciona el problema, pero le faltan unas líneas. Complétalas, por favor.

```
01 static boolean isPal(String s) {
02 if(s.length() == 0 || s.length() == 1)
03 return ........;
04 if(.........)
05 return isPal(s.substring(1, s.length()-1));
06 //else
07 return false;
08 }
```

En Java, el método s.charAt(i) permite saber qué caracter hay en la posición i de la cadena s y s.substring(a,b) retorna una subcadena de s entre los índices a y b−1.

Completen, por favor, las líneas faltantes:

1. Completa la línea 3 .....
a. true
b. false
c. s
R\ a.True

2. Completa la línea 4 .....
a. s.substring(0,s.length()-1).equals(s.substring(s.length()-1, 0))
b. s.charAt(0) == (s.charAt(s.length()-1))
c. true
R\ b. s.charAt(0) == (s.charAt(s.length()-1))

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

4.6 *Pepito escribió el siguiente código usando recursión:*

```
private int b(int[ ] a, int x, int low, int high) {
 if (low > high) return -1;
 int mid = (low + high)/2;
 if (a[mid] == x) return mid;
 else if (a[mid] < x)
 return b(a, x, mid+1, high);
 else
 return b(a, x, low, mid-1);
}
```

¿Cuál ecuación de recurrencia describe el comportamiento del algoritmo anterior para el peor de los casos?
a) T(n)=T(n/2)+C
b) T(n)=2.T(n/2)+C
c) T(n)=2.T(n/2)+Cn
d) T(n)=T(n−1)+C

R\ d) T(n)=T(n−1)+C

4.7 *¿Qué calcula el algoritmo desconocido y cuál es la complejidad asintótica en el peor de los casos del algoritmo desconocido?*

```
01 public int desconocido(int[] a){
02 return aux(a, a.length-1); }
03
04 public int aux(int[] a, int n){
05 if(n < 1) return a[n];
06 else return a[n] + aux(a, n-1); }
```

Elija la respuesta que considere acertada:

a) La suma de los elementos del arreglo a y es
b) Ordena el arreglo a y es O(n.log n)
c) La suma de los elementos del arreglo a y es O(1)
d) El máximo valor de un arreglo a y es O(n)
e) La suma de los elementos del arreglo a y es O(n)

R\ e) La suma de los elementos del arreglo a y es O(n)

4.8 *En la vida real, la teoría de juegos ha demostrado ser muy útil en la inteligencia artificial moderna; por ejemplo, para hacer transferencia de estilo que permite generar obras de arte con el estilo de un pintor determinado. Para este parcial, considera un juego en el que un jugador puede ganar 3, 5 o 7 puntos en un solo turno. ¿De cuántas maneras puede el jugador obtener un total de T puntos? A continuación, algunos ejemplos:*

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

• Para T=10. Respuesta: 3 que son 5+5, 7+3, 3+7.
• Para T=2. Respuesta: 0.
• Para T=15. Respuesta: 8.

El algoritmo ways soluciona el problema, pero le faltan unas líneas, para poder resolver el ejercicio deberás completarlas.

```
01 int ways(int T){
02 //Caso(s) base(s).
03 .....
04 .....
05 int f1 = ways(T - 3);
06 int f2 = ways(T - 5);
07 int f3 = ways(T - 7);
08 return .....;
09 }
```

4.5.1 ¿Cuántas instrucciones ejecuta el algoritmo en el peor de los casos?:
a. T(n)=T(n−1)+C
b. T(n)=T(n−1)+T(n−2)+C
c. T(n)=T(n/2)+C
d. T(n)=T(n+1)+C

R\ b. T(n)=T(n−1)+T(n−2)+C

***4.6** Alek y Krish están jugando Número. Número es un juego en el que un jugador 1, entrega un numero n (1<=n<=10100) a un jugador 2 y el jugador 2 debe determinar la suma de todos los dígitos de n, exceptuando el caso en el que hay dos dígitos adyacentes (es decir, contiguos, seguidos) que son iguales.*
Si hay dos dígitos adyacentes, no se suma ninguno de los dos números adyacentes. Entre Alek y Krish escribieron un código para hacer esto más rápido, pero se ha borrado una parte. ¿Podrían ayudarles a reconstruir el código a Alek y Krish?

```
1 public int suma(String n) {
2 return sumaAux(n, 0);
3 }
4
5 private int sumaAux(String n, int i){
6 if (i >= n.length()) {
7 return 0;
8 }
9 if(i + 1 < n.length() &&
 n.charAt(i) == n.charAt(i + 2)){
10 return sumaAux(n,i+1) ;
11 }
12 return (n.charAt(i) - '0') + sumaAux(n,i+1);
13 }
```

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

La operación n.charAt(i) - '0' convierte un caracter en su equivalente en entero, por ejemplo, el caracter '1' lo transforma en el número 1.

4.6.1 Completen la línea 10:

_____

4. 6.2 Completen la línea 12:

_____

***4.8** Considere el siguiente programa. ¿Cuál es la salida generada por fun(11,5)? Como un ejemplo: fun(10,3)=20.*

```
int fun(int n, int m){
 if(n % m == 2) return n;
 return fun(n + m, n - m);
}
```

Elija la respuesta que considere acertada:
a. 11
b. 5
c. 22
d. 2

R\ c. 22

***4.9** Considere el siguiente programa. ¿Cuál es la salida para fun(1,4)? Como un ejemplo: fun(1,2)=4.*

```
int fun(int m,int n){
 if(m==0){
 return (n+1);
 }
 if(m>0 && n==0){
 return fun(m-1,1);
 }
 int a=fun(m,n-1);
 return fun(m-1,a);
}
```

Elija la respuesta que considere acertada:
a. 4
b. 6
c. 5

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

d. 12

R\ b. 6

4.10 *Lika y Kefo están estudiando para el examen de matemáticas en su colegio. Hoy, ellos encontraron muchas secuencias de números interesantes, una de ellas los números Fibonacci. Para Lika –que ya conocía estos números– se le hace aburrido escribirlos todos; sin embargo, ellos encontraron otra secuencia de números llamados los números de Lucas. En el libro donde ellos encontraron esta secuencia de números, sólo hay 7 números pertenecientes a la secuencia y al final de la hoja dice: "¿Podrás definir una relación de recurrencia que nos permita deducir el n-ésimo número de Lucas?" "Por supuesto" –dijeron ambos. Ahora, ellos quieren escribir un algoritmo que nos permita retornar el n-ésimo número de Lucas. ¿Puedes ayudarlos a escribir el algoritmo? La secuencia encontrada fue la siguiente: 2, 1, 3, 4, 7, 11, 18, …. Se sabe que el número 2 y el número 1 son el primer y segundo término, respectivamente, de la secuencia de los números de Lucas. Al código le faltan algunas líneas, para completar el ejercicio deberás completarlas.*

```
1 int lucas(int n){
2 if(n == 0) return 2;
3 if(n == 1) return 1;
4 return lucas (return lucas(n-1) + lucas(n-2);
5 }
```

4.11.1 La complejidad asintótica del algoritmo anterior, para el peor de los casos, en términos de n, es:

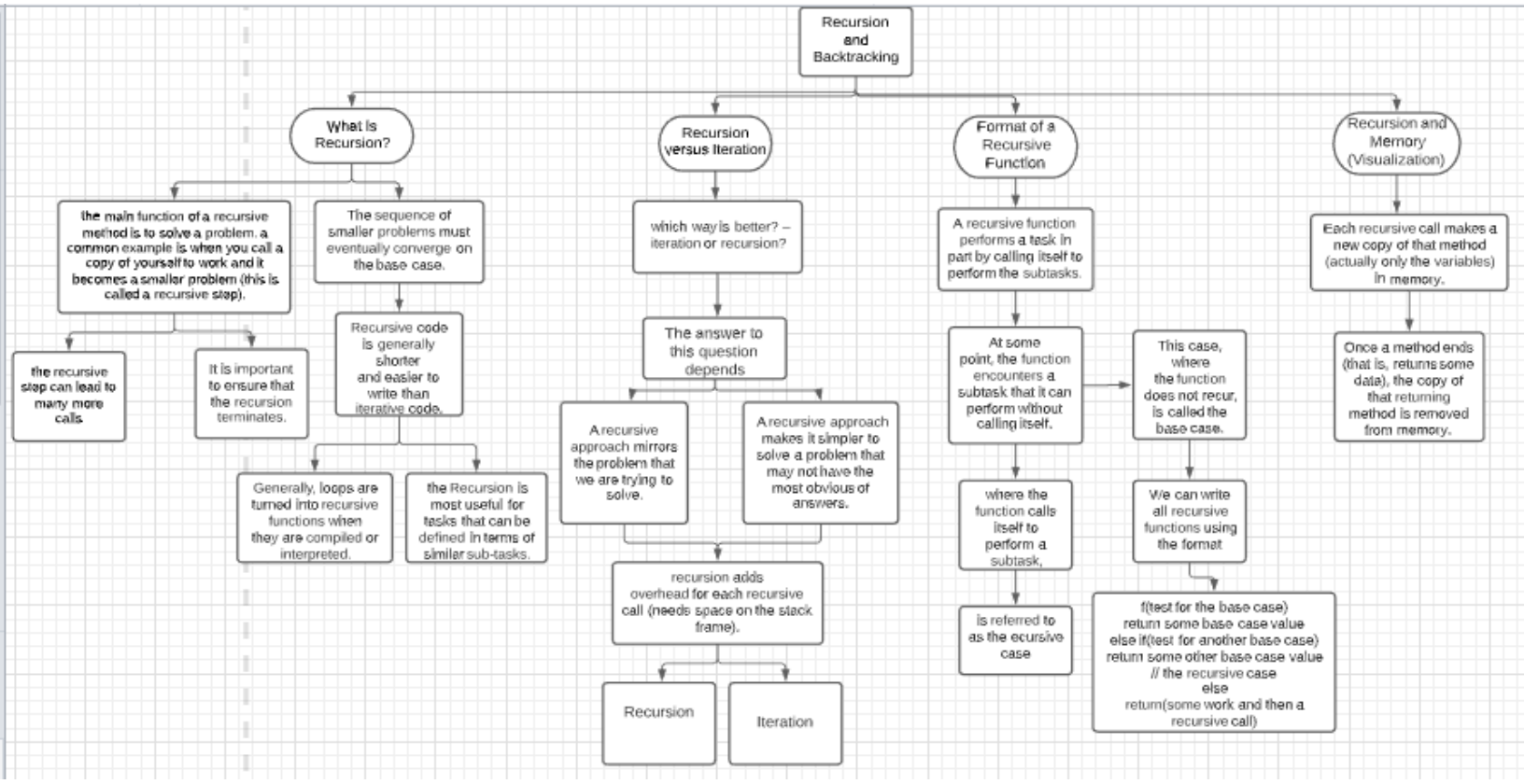
a. $T(n)=T(n-1)+c$, que es $O(n)$
b. $T(n)=4T(n/2)+c$, que es $O(n2)$
c. $T(n)=T(n-1)+T(n-2)+c$, que es $O(2n)$
d. $T(n)=c$, que es $O(1)$

R\ c. $T(n)=T(n-1)+T(n-2)+c$, que es $O(2n)$

5) Lectura recomendada (opcional)

Mapa conceptual

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473



https://app.lucidchart.com/invitations/accept/44432a03-c4dd-43cf-901c-70efaaea5545

6) Trabajo en Equipo y Progreso Gradual (Opcional)

- ***6.1*** *Actas de reunión*
- ***6.2*** *El reporte de cambios en el código*
- ***6.3*** *El reporte de cambios del informe de laboratorio*

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473