ESTRUCTURA DE DATOS 1
Código ST0245

Laboratory practice No. 2: Algorithm complexity

Miguel Ángel Martínez Flórez
Universidad Eafit
Medellín, Colombia
mamartinef@eafit.edu.co

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

3) Practice for final project defense presentation

3.1 Tables

Insertion sort		
Size:	Time:	Increase:
100000	10050	0
110000	12081	2031
120000	14956	2875
130000	17189	2233
140000	20134	2945
150000	23165	3031
160000	26298	3133
170000	29877	3579
180000	33246	3369
190000	36982	3736
200000	41296	4314
210000	45497	4201
220000	49960	4463
230000	54531	4571
240000	60052	5521
250000	64942	4890
260000	69958	5016
270000	75589	5631
280000	81262	5673
290000	88692	7430
300000	93540	4848

Merge sort		
Size:	Time:	Increase:
100000000	9918	0
105000000	10006	88
110000000	10124	118
115000000	10549	425
120000000	12607	2058
125000000	11420	-1187
130000000	11924	504
135000000	12911	987
140000000	13044	133
145000000	14072	1028
150000000	14112	40
155000000	15355	1243
160000000	15130	-225
165000000	15924	794
170000000	15997	73
175000000	17179	1182
180000000	17652	473
185000000	17602	-50
190000000	18677	1075
195000000	18612	-65
200000000	19555	943

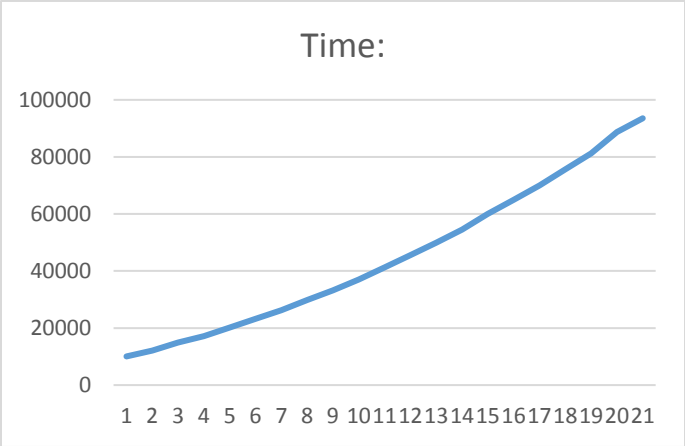
PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473

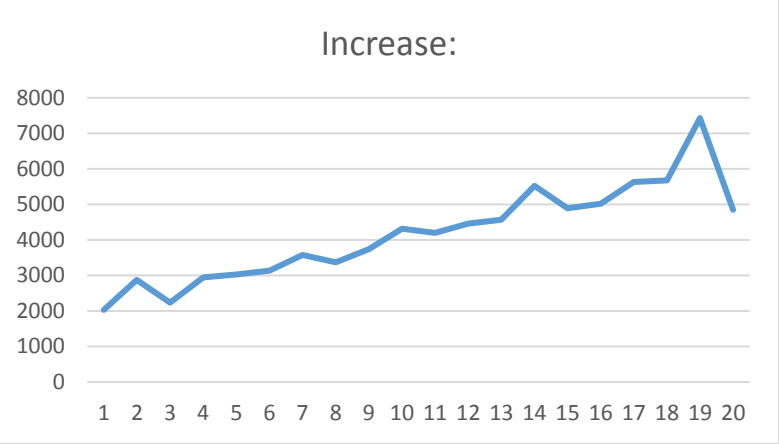
UNIVERSIDAD EAFIT® Acreditación Institucional Renovación 2018-2026 Resolución MEN 2158 de 2018

Inspira Crea Transforma | Vigilada Mineducación | www.eafit.edu.co

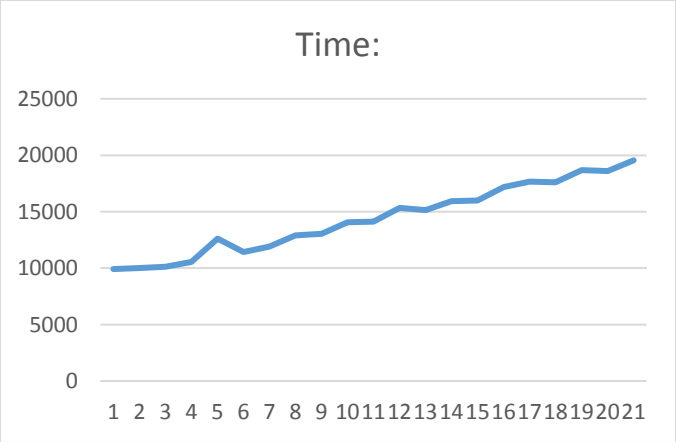
3.2 Graphs

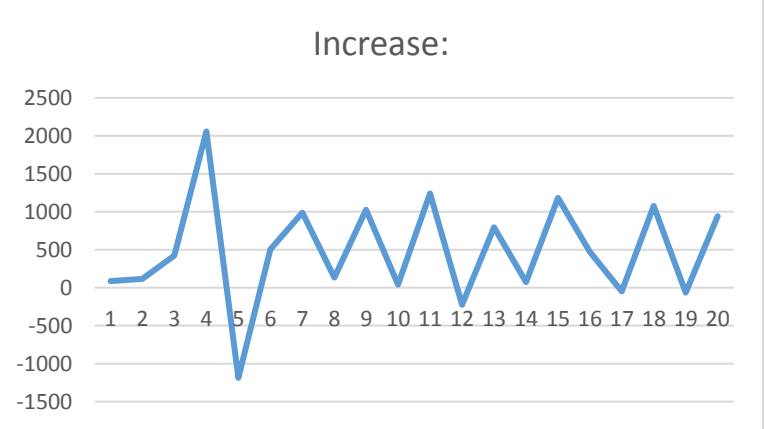
Insertion sort





Merge sort





3.3 Is it correct to use insertion sort with complex cases?

The complexity of insertion sort is 2^X in the worst case, this makes it so that with millions of numbers is very likely to find unfavorable cases which dramatically increases its delay time, it doesn't take much memory space but if you plan to use it in a situation where the response time should be 30-60 times per second with millions of elements, this algorithm is not sufficient.

3.4 Why does a logarithm appear in the complexity of merge sort?

The logarithm functions works as the exact opposite of 2^X (insertion sort's complexity), because the second one keeps getting bigger faster the more X increases, but logarithm grows slower the more X increases, this doesn't mean that 3 is faster than 2, it means that the increase in time is less each time, for example if it takes 2 = 10sec and 3 = 15sec the next increase will be less 4 = 18sec, four still takes more time but it is much more proficient at handling large numbers.

3.5 When is insertion sort superior?

PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473

Insertion sort in the best case scenario is the fastest way possible to organize an array, the best case is one in which the array is already organized, so the algorithm only checks each number once and its done with a complexity of n.

3.6 How does maxSpan functions?

maxSpan wants to know the maximum amount of elements inside 2 identical ones, so it checks each element and for each it cycles until the end, in this cycle it keeps count of the amount of elements it has passed, and if it reaches an element which is equal to the starter, then it saves the current element count as a possible maximum, after its done it compares the possible maximums and returns the biggest one.

3.7 Complexity of the online exercises.

Array 2:
countEvens T(n)= 2 + 4n = O(n)
centeredAverage T(n) = 4 + 6n + 2 = O(n)
sum67 T(n) = 3 + 6n = O(n)
more14 T(n) = 3 + 5n + 1 = O(n)
zeroMax T(n) = 1 + 4n + 5m = O(nm)
Array 3:
maxSpan T(n) = 5 + 5n + 5m = O(nm)
canBalance T(n) = 3 + 5n + 5m = O(nm)
linearIn T(n) = 3 + 3n + 5m + 1 = O(nm)
squareUp T(n) = 4 + 3n + 3m = O(nm)
seriesUp T(n) = 4 + 3n + 4m = O(nm)

3.8 Explain in your own words what are n and m.

The n is used when it is a cycle that is going to be repeated the same number of times as the entered value or size of the entered value, such as a method that receives an n and has a cycle that does n times a statement would have a complexity of n.
The m is used when it is a cycle that repeats a number of times of a value created by n, that is to say, another cycle that takes its complexity from a value created by n; for example if there are 2 separate cycles, the first cycle creates a variable and the second makes its cycle based on that one, then it would be a complexity of n + m, or another more common example is a cycle within another cycle, where m uses the derivatives of each n to make its cycles and repeats with each cycle of n, there it would have a complexity of n*m.

4) Practice for midterms

4.1 Supongamos que P(n,m) es una función cuya complejidad asintótica es O(n×m) y H(n,m) es otra función cuya complejidad asintótica es O(m*A(n,m)), donde A(n,m) es otra función.

¿Cuál de las siguientes funciones, definitivamente, NO podría ser la complejidad asintótica de A(n,m) si tenemos en cuenta que P(n,m)>H(n,m) para todo n y para todo m?

PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473

Elija la respuesta que considere acertada:
a) O(log n)
b) O(√n)
c) O(n+m)
d) O(1)

R\ c) O(n+m)

4.2 Dayla sabe que la complejidad asintótica de la función P(n) es O(√n). Ayúdenle a Dayla a sacar la complejidad asintótica para la función mistery(n,m).

```
void mistery(int n, int m) {
 for(int i = 0; i < m; ++i){
 boolean can = P(n);
 if(can)
 for(int i = 1; i * i <= n; i++)
 //Hacer algo en O(1)
 else
 for(int i = 1; i <= n; i++)
 //Hacer algo en O(1)
} }
```

La complejidad de mistery(n,m) es:
a) O(m+n)
b) O(m×n×√n)
c) O(m+n+√n)
d) O(m×n)

R\ b) O(m×n×√n)

4.3 El siguiente algoritmo imprime todos los valores de una matriz. El tamaño de la matriz está definido por los parámetros largo y ancho. Teniendo en cuenta que el largo puede ser como máximo 30, mientras que el ancho puede tomar cualquier valor, ¿Cuál es su complejidad asintótica en el peor de los casos del algoritmo?

```
01public void matrices(int[][] m, int largo, int ancho)
02 for(int i=0; i < largo; i++)
03 for(int j=0; j < ancho; j++)
04 print (m[i][j]);
```

Elija la respuesta que considere acertada:

PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473

a) O(largo)
b) O(ancho)
c) O(largo+ancho)
d) O(ancho×ancho)
e) O(1)

R\ b) O(ancho)

4.4 Sabemos que P(n) ejecuta n 3 + n pasos y que D(n) ejecuta n+7 pasos. ¿Cuál es la complejidad asintótica, en el peor de los casos, de la función B(n)?

```
public int B(int n)
 int getP = P(n);
 int ni = 0;
for(int i = 0; i < n; ++i)
 if(D(i) > 100){
 ni++;
 int nj = getP + D(n) * ni;
 return nj;
```

Elija la respuesta que considere acertada:
a) O(n4)
b) O(n3)
c) O(n2)
d) O(2n)

R\ a) O(n4)

4.5 Considera el siguiente algoritmo:

```
static int count7(int n) {
 if (n == 0) return 0;
 if (n % 10 == 7) return 1 + count7(n / 10);
 return count7(n / 10);
}
```

¿Cuál es la ecuación de recurrencia que mejor define la complejidad, para el peor caso, del algoritmo anterior? Asume que c es la suma de todas las operaciones que toman un tiempo constante en el algoritmo.

(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)=T(n/10)+c, que es O(logn)

R\ (d)T(n)=T(n/10)+c, que es O(logn)

PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473

¿El algoritmo anterior siempre termina para todo número entero n∈Z?

(a)Sí
(b)No

R\(b)No

4.6 Juanito implementó un algoritmo para sumar 2 matrices cuadradas de dimensión N. Su algoritmo tiene complejidad T(n)=c×n 2 y toma T(n) segundos para procesar n datos.

¿Cuánto tiempo tardará este algoritmo para para procesar 10000 datos, si sabemos que, para n=100, T(n)=T(100)=1? Recuerda que 1=1000. Así como en los parciales de Física 1, NO olvides indicar la unidad de medida del tiempo que calcules.

R\ 100s, debido a que cuando n es 100 se demora 1 segundo podemos despejar la complejidad para encontrar el valor de c la cual se descubre que es 1/1millon, ya simplemente utilizamos la complejidad para revisar cuanto se demora a lo cual llegamos a 100.000.000/1.000.000 lo que resulta en 100 y se denota a segundos (s)

4.7 Considera las siguientes proposiciones:

1. O(f+g)=O(max(f,g))
2. O(f×g)=O(f)×O(g)
3. Si f=O(g) y g=O(h), entonces f=O(h)
4. O(c.f)=O(f), donde c es una constante

¿Cuál(es) de las anteriores proposiciones son verdaderas?
1. O(f+g)=O(max(f,g))
3. Si f=O(g) y g=O(h), entonces f=O(h)
4. O(c.f)=O(f), donde c es una constante

4.8 ¿Cuál de las siguientes afirmaciones es correcta con respecto a func3(n)?

```
1 void func3(int n){
2 if(n < 1) return;
3 else{
4 System.out.println(n);
5 func3(n-1);
6 }
7 }
```

Elija la respuesta que considere acertada:

PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473

a) Esta ejecuta $T(n) = c + T(n-1)$ pasos, que es O(n).
b) Esta ejecuta $T(n) = n + T(n-1)$ pasos, que es O($n!$).
c) Esta ejecuta $T(n) = cn + T(n-1)$ pasos, que es O($n!$).
d) Esta ejecuta $T(n) = c + 2.\ T(n-1)$ pasos, que es O($2n$).

R\ a) Esta ejecuta $T(n) = c + T(n-1)$ pasos, que es O(n).

4.9 Considera el siguiente algoritmo:

```
void f(int n){
 for(int i=1; i*i <= n; i++) {
 for(int j=1; j*j<=n; j++) {
 for(int k=0; k<n; k++) {
 for(int h=0; h<=n; h++) {
 System.out.println("hola");
 }
 }
 }
 }
}
```

¿Cuál es la complejidad asintótica, para el peor de los casos, del algoritmo f(n) para n>1?

a) O(n3)
b) O(n2)
c) O(n3× n)
d) O(n4× n)

R\ a) O(n3)

4.10 ¿Cuál de las siguientes afirmaciones es correcta con respecto a la función func2(n)?

```
void func2(int n){
 for(int i = 2; i * i <= n; i++){
 for(int k = 2; k * k <= n; k++){
 print(j);
 }
 }
}
```

Elija la respuesta que considere acertada:
a) Ejecuta más de n 2 pasos.
b) Ejecuta más de n 3 pasos.
c) Ejecuta menos de n.log n pasos.

PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473

d) Ejecuta exactamente n 2 pasos

R\ d) Ejecuta exactamente n 2 pasos

4.11 ¿Cuál de las siguientes afirmaciones es correcta con respecto a func3(n)?

```
int func3(int n){
 if(n == 1 || n == 2){
 return n;
 }
 int ni = func3(n - 1);
 int nj = func3(n - 2);
 int suma = ni + nj;
 return suma;
}
```

Elija la respuesta que considere acertada:
a) Ejecuta T(n)=T(n−1)+c pasos.
b) Ejecuta T(n)=T(n−1)+cn pasos.
c) Ejecuta T(n)=T(n−1)+T(n−2)+c pasos.
d) Ejecuta T(n)=T(n/2)+c pasos.

R\ c) Ejecuta T(n)=T(n−1)+T(n−2)+c pasos.

4.12 Sea $f(n, m) = n \times \log(n) + m2$ y $g(n, m) = n + m$. Calcule O($f(n, m) \times g(n, m)$). Ten en cuenta que si la regla del producto fuera válida para el producto de variables, entonces O($n \times n$) sería O(n), lo cual no es cierto. Ten en cuenta que no se sabe cuál es más grande entre n y m.

Elija la respuesta que considere acertada:
a) O($n \times \log(n) + m2$)
b) O($m \times n \times \log(n) + n \times m2 + n \times \log(n) + m3$)
c) O($m3$)
d) O($n3 + m3$)

R\ b) O($m \times n \times \log(n) + n \times m2 + n \times \log(n) + m3$)

4.13 Considere el siguiente código escrito en Java. Encuentre la ecuación de recurrencia que mejor representa la complejidad asintótica en el peor de los casos.

```
int f(int n){
 if(n <= 0){
 return 1;
 }
```

PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473

```
 int a = f(n / 2);
 int b = f(n / 2);
 int res = 0;
 for(int i = 0; i < n; i++){
 res += (a*b);
 }
 return res;
}
```

Elija la respuesta que considere acertada:
a) T(n)=2T(n−1)+n
b) T(n) = 2T(n/2) + n2
c) T(n) = 2T(n/2) + n
d) T(n)=2T(n−1)+(n−1)

R\ c) T(n) = 2T(n/2) + n

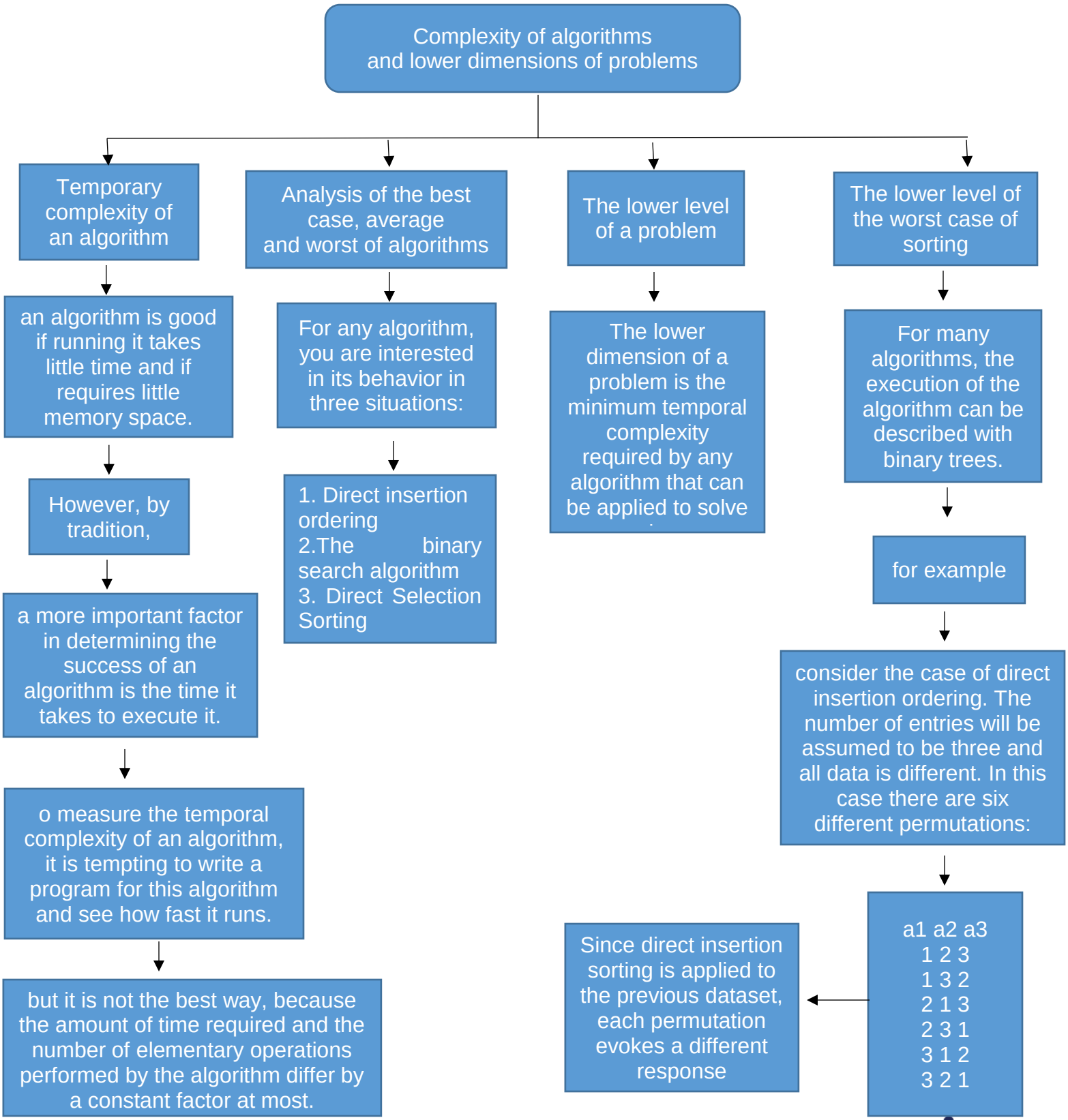
4.14 Sea $f(n, m) = n\ 2 + n \times \log(log(m))$ y $g(n, m) = n\ 3 + m \times \sqrt{m}$. Calcule $O(f(n, m) + g(n, m))$. Ten en cuenta que no se sabe cuál es más grande entre n y m.

Elija la respuesta que considere acertada:
a) $O(n3 + n(\log(\log(m)) + m \times \sqrt{m})$
b) $O(n3)$
c) $O(m \times \sqrt{(m)} + n3)$
d) $O(m \times \sqrt{m})$
R\ a) $O(n3 + n(\log(\log(m)) + m \times \sqrt{m})$ o la c) $O(m \times \sqrt{(m)} + n3)$

5) Recommended reading (optional)

PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473

ESTRUCTURA DE DATOS 1
Código ST0245



PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473

UNIVERSIDAD EAFIT®
Acreditación Institucional
Renovación 2018 - 2026
Resolución MEN 2158 de 2018

Inspira Crea Transforma
Vigilada Mineducación
www.eafit.edu.co