INSTITUTO POLITECNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO (ESCOM)

ANÁLISIS DE ALGORITMOS

NOMBRE DEL ALUMNO:

- SANTOS MÉNDEZ ULISES JESÚS

EJERCICIO 07:

- ANÁLISIS DE ALGORITMOS NO RECURSIVOS

FECHA DE ENTREGA:

- 15/04/2022

GRUPO:

- 3CM14

C

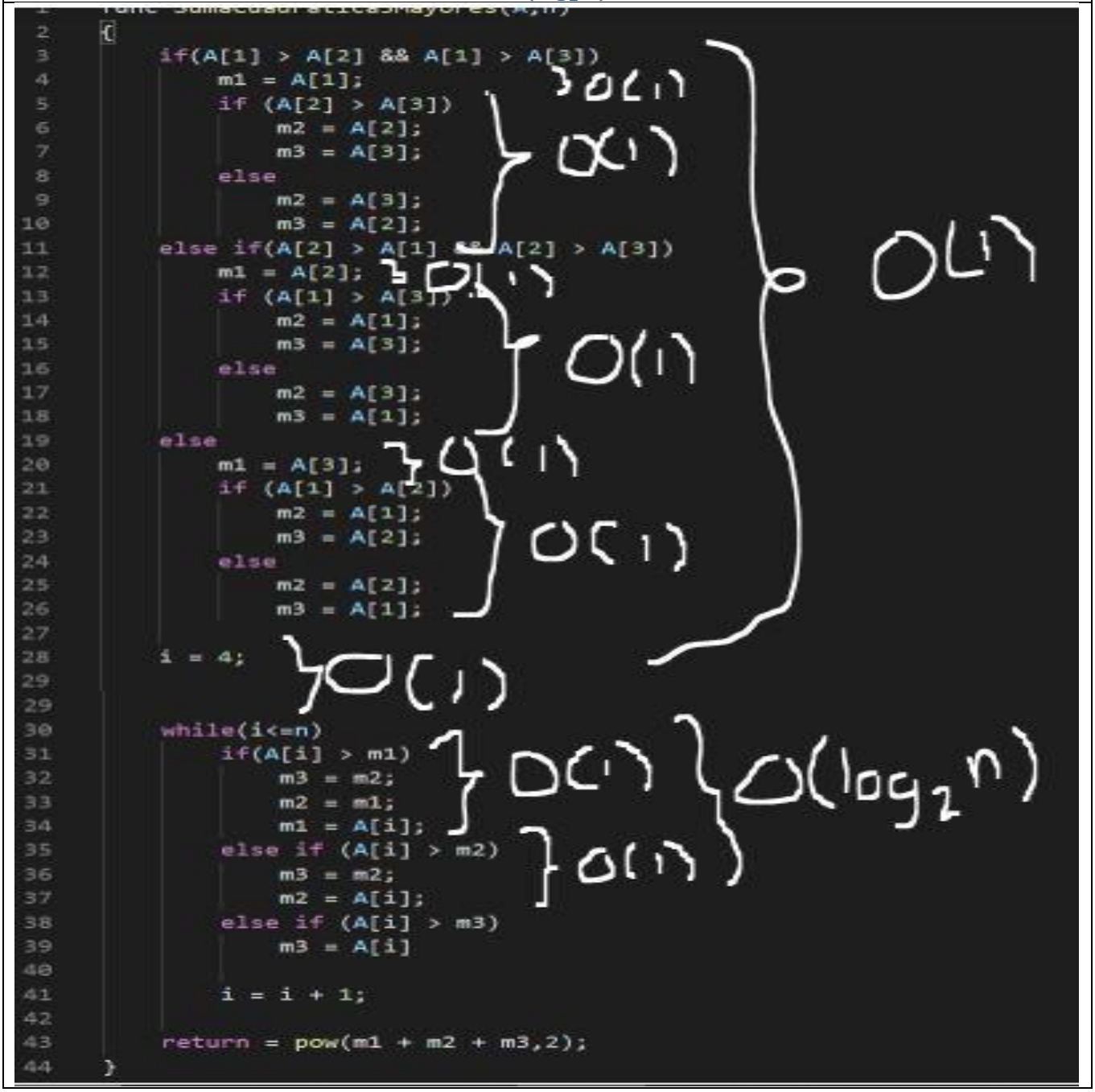
Análisis de algoritmos no recursivos

Objetivo:

Para los siguientes algoritmos determine la cota de complejidad O(cota superior ajustada) bajo el principio del peor caso, de cada algoritmo, Indique a detalle el procedimiento de obtención de la cota.

Código 01:

$\in O(\log_2 n)$



Código 02:

$\in O(n^2)$

```
1   func OrdenamientoIntercambio(A,n)
2   for (i=0; i<n-1; i++)
3       for ( j=i+1; j<n;j++)
4           if (A[j] < A[i])
5           {
6               temp=A[i];
7               A[i]=A[j];
8               A[j]=temp;
9           }
10  fin
```

Código 03:

$\in O(\log_2 n)$

```
1   func MaximoComunDivisor(m, n)
2   {
3       a=max(n,m);
4       b=min(n,m);
5       residuo=1;
6       mientras (residuo > 0)
7       {
8           residuo=a mod b;
9           a=b;
10          b=residuo;
11      }
12      MaximoComunDivisor=a;
13      return MaximoComunDivisor;
14  }
```

Código 04:

$\in O(n^2)$

Código 04:

```
1   func SumaCuadratica3MayoresV2(A,n)
2   {
3       for(i=0;i<3;i++)
4       {
5           for (j=0;j<n-1-i;j++)
6           {
7               if(A[j]>A[j+1])
8               {
9                   aux=A[j];
10                  A[j]=A[j+1];
11                  A[j+1]=aux;
12              }
13          }
14      }
15      r=A[n-1] + A[n-2] + A[n-3];
16      return pow(r,2);
17  }
```

Código 05:

$\in O(n \log n)$

```
1   Procedimiento BurbujaOptimizada(A,n)
2       cambios = "Si"
3       i=0
4       Mientras i< n-1 && cambios != "No" hacer
5           cambios = "No"
6           Para j=0 hasta (n-2)-i hacer
7               Si(A[j] < A[j]) hacer
8                   aux = A[j]
9                   A[j] = A[j+1]
10                  A[j+1] = aux
11                  cambios = "Si"
12              FinSi
13          FinPara
14           i= i+1
15      FinMientras
16  fin Procedimiento
```

Código 06:

$\in O(n^2)$

```
1   Procedimiento BurbujaSimple(A,n)
2       para i=0 hasta n-2 hacer
3           para j=0 hasta (n-2)-i hacer
4               si (A[j]>A[j+1]) entonces
5                   aux = A[j]
6                   A[j] = A[j+1]
7                   A[j+1] = aux
8               fin si
9           fin para
10      fin para
11  fin Procedimiento
```

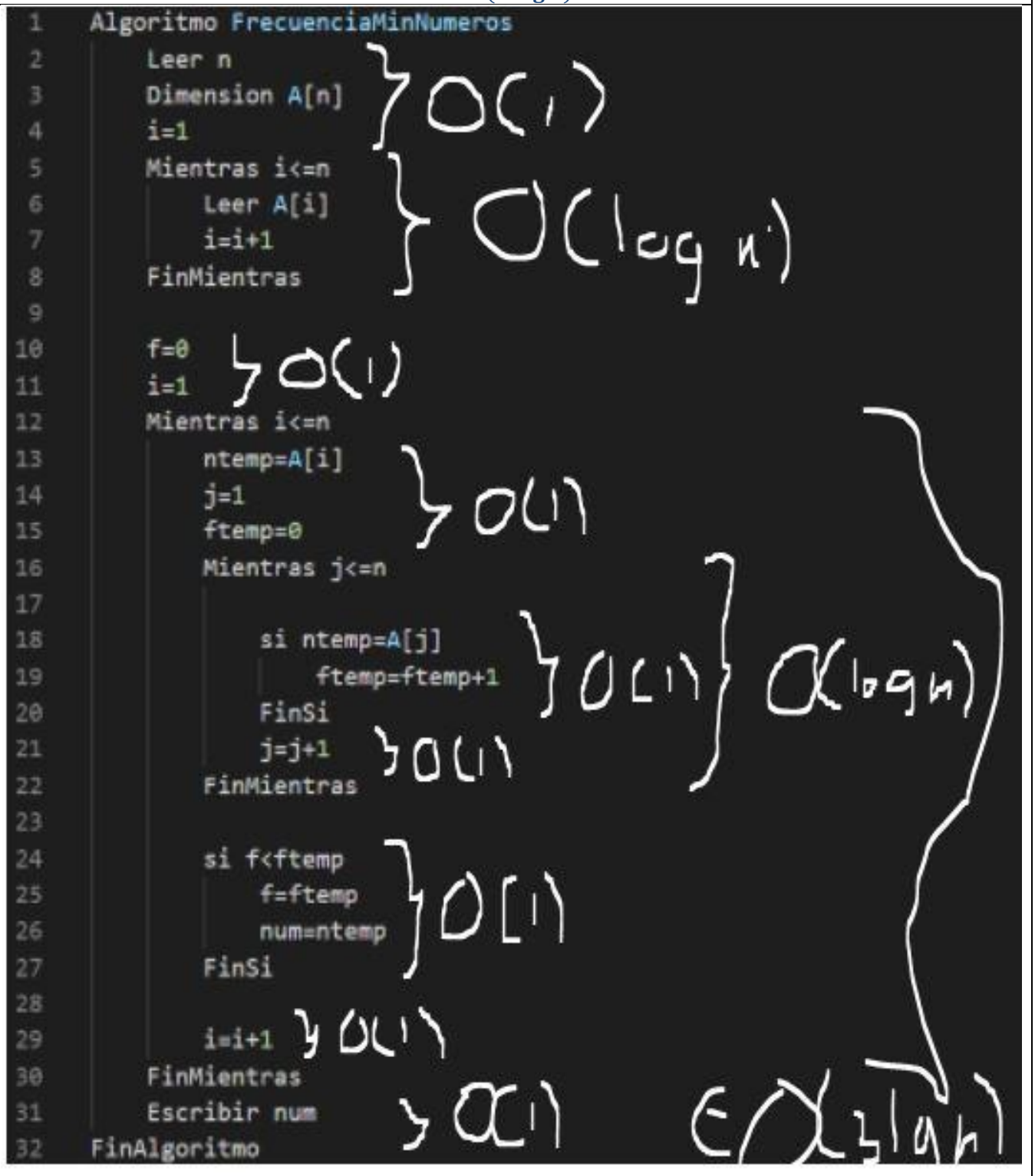
Código 07:

$\in O(n)$

```
1   Proceso ValidaPrimo
2       Leer n
3       divisores<-0
4       si n>0 Entonces
5           Para i<-1 Hasta n Hacer
6               si (n%i=0) Entonces
7                   divisores=divisores+1
8               FinSi
9           FinPara
10      FinSi
11
12      si divisores=2
13          Escribir 'S'
14      SiNo
15          Escribir 'N'
16      FinSi
17  FinProceso
```

Código 08:

$\in O(3 \log n)$



Código 09:

$\in O((N-M)M)$

```
1   void search(char* pat, char* txt)
2   {
3       int M = strlen(pat);
4       int N = strlen(txt);
5
6       for (int i = 0; i <= N - M; i++)
7       {
8           int j;
9
10          for (j = 0; j < M; j++)
11              if (txt[i + j] != pat[j])
12                  break;
13
14          if (j == M)
15              printf("Pattern found at index %d \n", i);
16      }
17  }
```

Código 10:

$\in O(n \log n)$

```
1   stack<int> sortStack(stack<int> &input)
2   {
3       stack<int> tmpStack;
4
5       while (!input.empty())
6       {
7           int tmp = input.top();
8           input.pop();
9
10          while (!tmpStack.empty() && tmpStack.top() > tmp)
11          {
12              input.push(tmpStack.top());
13              tmpStack.pop();
14          }
15
16          tmpStack.push(tmp);
17      }
18      return tmpStack;
```