Instituto Politécnico Nacional

Escuela Superior de Cómputo

COMPLEJIDAD TEMPORAL Y ANÁLISIS DE CASOS

Ejercicio 02

Análisis de Algoritmos

M. en C. Edgardo Adrián Franco Martínez
Grupo: 3CM3
Fecha: 11 /Marzo/2018

Alumno:

Calva Hernández José Manuel 2017630201

Índice

Sección A **2**

Algoritmo 1 2

Algoritmo 2 2

Algoritmo 3 3

Algoritmo 4 3

Algoritmo 5 4

Sección B **5**

Algoritmo 6 5

Algoritmo 7 6

Algoritmo 8 7

Sección C **9**

Algoritmo 9 9

Algoritmo 10 10

Algoritmo 11 11

Algoritmo 12 12

Algoritmo 13 13

Algoritmo 14 14

Algoritmo 15 15

Sección A.

Para los siguientes 5 algoritmos determine la función de complejidad temporal y espacial en términos de n.
*Considere las operaciones de: asignación, aritméticas, condicionales y saltos implícitos.

Algoritmo 1

```
for(i = 1;i<n;i++){              ->  1 (asignación) + n (comparaciones) + 2(n-1) (asignaciones)
    for(j=0;j<n-1;j++){          ->  (n-1) ( 1 (asignación) + n (comparaciones) + 2(n-1) (asignaciones) )
        temp = A[j];             ->  (n-1)² ( 1 (asignación) )
        A[j] = A[j+1];           ->  (n-1)² ( 1 (asignación) + 1 (adición) )
        A[j+1] = temp;           ->  (n-1)² ( 1 (asignación) + 1 (adición) )
    }                            ->  n-1 ( n-1 (saltos implícitos) + 1 (salto en falso) )
}                                ->  n-1 (saltos implícitos) + 1 (salto en falso)
```

Realizando las operaciones, obtenemos la siguiente fórmula temporal:

$$f_t(n) = 9n^2 - 12n + 5$$

Dado que se utiliza una variable auxiliar para guardar los elementos a intercambiar, además de las dos variables para los ciclos y el arreglo que presuntamente tiene tamaño n, la fórmula espacial es:

$$f_e(n) = n + 3$$

Algoritmo 2

```
polinomio = 0;                          ->  1 (asignación)
for (int i = 0; i <= n; ++i)            ->  1 (asignación) + n+2 (comparaciones) + 2(n+1) (asignaciones)
{
    polinomio = polinomio * z + A[n-i];  -> n+1 ( 1 (producto) + 1 (adición) + 1 (sustracción) + 1 (asignación) )
}                                        -> n+1 (saltos implícitos) + 1 (salto en falso)
```

Realizando las operaciones, obtenemos la siguiente fórmula temporal:

$$f_t(n) = 8n + 12$$

Tomaremos en cuenta un arreglo presuntamente de tamaño n, además de una variable auxiliar para ir llevan el resultado, una variable multiplicativa "z" y la variable utilizada para el ciclo; por tanto la fórmula espacial es:

$$f_e(n) = n + 3$$

Algoritmo 3

```
1  for i = 1 to n do                                  ->  1 (asignación) + n+1 (comparaciones) + 2n (asignaciones)
2  {
3      for j = 1 to n do{                             ->  n ( 1 (asignación) + n+1 (comparaciones) + 2n (asignaciones) )
4          c[i,j] = 0;                                ->  n² ( 1 (asignación) )
5          for k = 1 to n do{                         ->  n² ( 1 (asignación) + n+1 (comparaciones) + 2n (asignaciones) )
6              c[i,j] = c[i,j] + A[i,k] * B[k,j];     ->  n³ ( 1 (producto) + 1 (adición) + 1 (asignación) )
7          }                                          ->  n² ( n (saltos implícitos) + 1 (saltos en falso) )
8      }                                              ->  n ( n (saltos implícitos) + 1 (salto en falso) )
9  }                                                  -> n (saltos implícitos) + 1 (salto en falso)
```

Realizando las operaciones, obtenemos la siguiente fórmula temporal:

$$f_t(n) = 7n^3 + 8n^2 + 7n + 3$$

Se utilizarán tres variables para los ciclos, además, se tienen tres arreglos bidimensionales, presuntamente de tamaño n*n, por tanto, la fórmula espacial es:

$$f_e(n) = 3n^2 + 3$$

Algoritmo 4

```
1  anterior = 1;                          ->  1 (asignación)
2  actual = 1;                            ->  1 (asignación)
3  while(n>2){                            ->  n-1 (comparaciones)
4      aux = anterior + actual;           ->  (n-2) ( 1 (asignación) + 1 (adición) )
5      anterior = actual;                 ->  (n-2) ( 1 (asignación) )
6      actual = aux;                      ->  (n-2) ( 1 (asignación) )
7      n = n-1;                           ->  (n-2) ( 1 (asignación) + 1 (sustracción) )
8  }                                      ->  n-2 (saltos implícitos) + 1 (salto en falso)
```

Realizando las operaciones, obtenemos la siguiente fórmula temporal:

$$f_t(n) = 8n - 12$$

Únicamente se utilizarán tres variables durante la ejecución del código, por tanto, la fórmula espacial es:

$$f_e(n) = 3$$

Algoritmo 5

```
for (int i = n-1, j = 0; i >= 0; --i, j++)      ->  1 (asignación) + 1(asignación) + n+1 (comparaciones) + 2n (asignaciones) + 2n (asignaciones)
{
    s2[j] = s[i];                               ->  n ( 1 (asignaciones) )
}                                               ->  n (saltos implícitos) + 1 (salto en falso)
for (int i = 0; i < n; ++i)                     ->  1 (asignación) + n+1 (comparaciones) + 2n (asignaciones)
{
    s[i] = s2[i];                               ->  n ( 1 (asignación) )
}                                               ->  n (saltos implícitos) + 1 (salto en falso)
```

Realizando las operaciones, obtenemos la siguiente fórmula temporal:

$$f_t(n) = 12n + 7$$

Se utiliza una misma variable para ambos ciclos, pero en el primero se tiene otra variable adicional, además, tenemos dos arreglos que asumiremos tienen de tamaño n, así, la fórmula espacial es la siguiente:

$$f_e(n) = 2n + 2$$

Sección B.

Para los siguientes 3 algoritmos determine el número de veces que se imprime la palabra “Algoritmos”. Determine una función lo más cercana a su comportamiento para cualquier n y compruébela codificando los tres algoritmos (Adjuntar códigos). De una tabla de comparación de sus pruebas para n’s igual a 10, 100, 1000, 5000 y 100000 y demostrar lo que la función encontrada determina será el número de impresiones.

Algoritmo 6

```c
#include <stdio.h>

int main(int argc, char const *argv[])
{
    int n, count = 0;
    scanf("%i",&n);
    for (int i = 10; i < n*5; i*=2)
    {
        count++;
        printf("\"Algoritmos\"\n");
    }
    printf("%i\n",count);
    return 0;
}
```

Dado que el ciclo irá aumentando al doble, se considera que este es exponencial. Por tanto, la función principal irá respecto de un logaritmo de base 2, y se considera únicamente la n porque la i está iniciándose en el doble de 5, que es 10. Se le hará función piso a la fórmula debido a que el ciclo nunca llegará a igualarse con el valor de n.

$$f(n) = \lfloor \log_2 n \rfloor$$

n	$f(n)$	# Impresiones Reales
10	$\lfloor \log_2 10 \rfloor = \lfloor 3.32 \rfloor = 3$	3
100	$\lfloor \log_2 100 \rfloor = \lfloor 6.64 \rfloor = 6$	6
1000	$\lfloor \log_2 1000 \rfloor = \lfloor 9.96 \rfloor = 9$	9
5000	$\lfloor \log_2 5000 \rfloor = \lfloor 12.28 \rfloor = 12$	12
100000	$\lfloor \log_2 100000 \rfloor = \lfloor 16.6 \rfloor = 16$	16

Algoritmo 7

```c
#include <stdio.h>

int main(int argc, char const *argv[])
{
    int n, count = 0;
    scanf("%i",&n);
    for (int j = n; j > 1; j/=2)
    {
        if (j<(n/2))
        {
            for (int i = 0; i < n; i+=2)
            {
                count++;
                printf("\"Algoritmos\"\n");
            }
        }
    }
    printf("%i\n",count);
    return 0;
}
```

La primera parte corresponde al ciclo interno, debido a que inicia en 0 y va aumentando de 2 en 2, se considera una función techo respecto a la mitad de n. Por otra parte, el ciclo externo se considera que irá reduciendo a la mitad la n, sin embargo, al tener la restricción de menor a su primera mitad y ser mayor a 1, descontaremos dos valores del resultado inicial que nos arroje la función piso sobre el logaritmo.

$$f(n) = \left(\left\lceil \frac{n}{2} \right\rceil\right)(\lfloor \log_2 n \rfloor - 2)$$

n	$f(n)$	# Impresiones Reales
10	$\left(\left\lceil \frac{10}{2} \right\rceil\right)(\lfloor \log_2 10 \rfloor - 2) = (5)(\lfloor 3.32 \rfloor - 2) = 5$	5
100	$\left(\left\lceil \frac{100}{2} \right\rceil\right)(\lfloor \log_2 100 \rfloor - 2) = (50)(\lfloor 6.64 \rfloor - 2) = 200$	200
1000	$\left(\left\lceil \frac{1000}{2} \right\rceil\right)(\lfloor \log_2 1000 \rfloor - 2) = (500)(\lfloor 9.96 \rfloor - 2) = 3500$	3500
5000	$\left(\left\lceil \frac{5000}{2} \right\rceil\right)(\lfloor \log_2 5000 \rfloor - 2) = (2500)(\lfloor 12.28 \rfloor - 2) = 25000$	25000
100000	$\left(\left\lceil \frac{100000}{2} \right\rceil\right)(\lfloor \log_2 100000 \rfloor - 2) = (50000)(\lfloor 16.6 \rfloor - 2) = 700000$	700000

Algoritmo 8

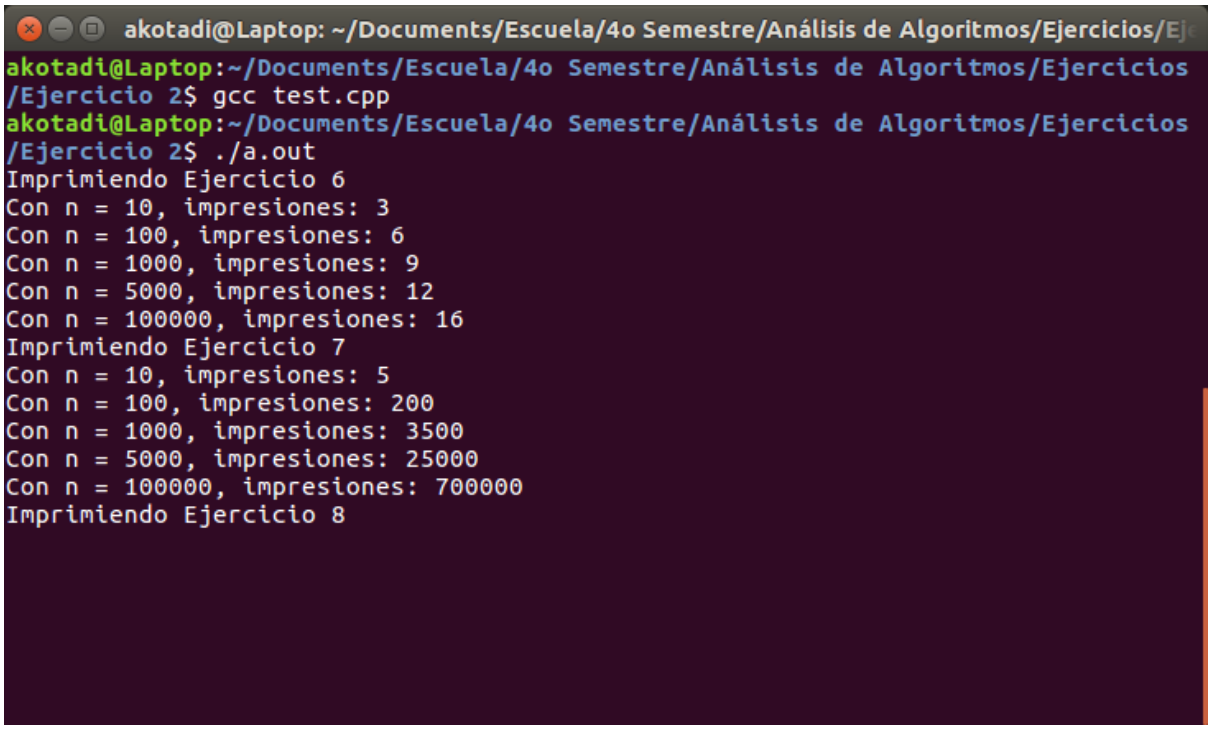
```c
#include <stdio.h>

int main(int argc, char const *argv[])
{
    int n, count = 0;
    scanf("%i",&n);
    int i = n;
    while(i>=0){
        for (int j = n; i < j; i-=2,j/=2)
        {
            count++;
            printf("\"Algoritmos\"\n");
        }
    }
    printf("%i\n",count);
    return 0;
}
```

Es sencillo ver que este ciclo será infinito siempre que la n sea mayor a 0, sin embargo, nunca va a imprimir nada debido a que la condición del ciclo interno nunca se podrá inicializar al ser tanto la i como la j del mismo valor que n.

$$f(n) = 0$$

n	$f(n)$	# Impresiones Reales
10	0	0
100	0	0
1000	0	0
5000	0	0
100000	0	0


akotadi@Laptop: ~/Documents/Escuela/4o Semestre/Análisis de Algoritmos/Ejercicios/Ej
akotadi@Laptop:~/Documents/Escuela/4o Semestre/Análisis de Algoritmos/Ejercicios
/Ejercicio 2$ gcc test.cpp
akotadi@Laptop:~/Documents/Escuela/4o Semestre/Análisis de Algoritmos/Ejercicios
/Ejercicio 2$ ./a.out
Imprimiendo Ejercicio 6
Con n = 10, impresiones: 3
Con n = 100, impresiones: 6
Con n = 1000, impresiones: 9
Con n = 5000, impresiones: 12
Con n = 100000, impresiones: 16
Imprimiendo Ejercicio 7
Con n = 10, impresiones: 5
Con n = 100, impresiones: 200
Con n = 1000, impresiones: 3500
Con n = 5000, impresiones: 25000
Con n = 100000, impresiones: 700000
Imprimiendo Ejercicio 8

Sección C.

Para los siguientes algoritmos determine las funciones de complejidad temporal, para el mejor caso, peor caso y caso medio. Indique cual(es) son las condiciones (instancia de entrada) del peor caso y cual(es) la del mejor caso.

Algoritmo 9

```
1  // Se tomarán en cuenta como operaciones la comparación en arreglo A, asignaciones mayor1 y mayor2
2
3  func Producto2Mayores(A,n){
4      if (A[1] > A[2])                              ->  1 (comparación)
5      {
6          mayor1 = A[1];                            ->  1 (asignación)
7          mayor2 = A[2];                            ->  1 (asignación)
8      }
9      else{
10         mayor2 = A[1];                            ->  1 (asignación)
11         mayor1 = A[2];                            ->  1 (asignación)
12     }
13     // Podemos observar que la primera comparación siempre va a realizar, y dado que el número de operaciones
14     // en ambos caminos es el mismo, tomaremos como valor 3 operaciones de este bloque de instrucciones
15     i = 3;
16     while(i<=n){                                  ->  n-2 (comparaciones)
17         if (A[i]>mayor1)                          ->  1 (comparación)
18         {
19             mayor2 = mayor1;                      ->  1 (asignación)
20             mayor1 = A[i];                        ->  1 (asignación)
21             // En este camino encontramos 3 operaciones en total
22         }
23         else if (A[i]>mayor2)                     ->  1 (comparación)
24         {
25             mayor2 = A[i];                        ->  1 (asignación)
26             // Por este, también encontramos 3 operaciones
27         }
28         // Si no se entra a ningún if, únicamente se realizan 2 operaciones
29         i++;
30     }
31     return = mayor1 * mayor2;
32 }
33 fin
```

Mejor Caso

Se da cuando los dos primeros elementos del arreglo son los mayores, ya que ello implica que nunca entrará a realizar las operaciones dentro de los ifs, por tanto, su fórmula sería:

$$f_t(n) = 2(n-2) + 3 = 2n - 1$$

Peor Caso

Lo encontramos cuando el arreglo está ordenado de menor a mayor, o bien, el primer o segundo elemento es uno de los mayores, pero los demás se encuentran ordenamos de forma ascendente. Esto implica que entrará en la parte del código de los ifs y realizará la totalidad de las operaciones posibles, por tanto, su fórmula sería:

$$f_t(n) = 3(n-2) + 3 = 3n - 3$$

Caso Medio

Como se puede observar en las anotaciones del código, encontramos 3 posibles caminos si no tomamos en cuenta la primer parte, debido a que las posibilidades dan el mismo resultado, lo cual no afectará nuestro

resultado. Por tanto, suponiendo que la probabilidad de que tome alguno de los tres posibles caminos del ciclo while es igual, consideraremos la siguiente fórmula:

$$f_t(n) = \frac{1}{3}(3n-3) + \frac{1}{3}(3n-3) + \frac{1}{3}(2n-1) = \frac{8n-7}{3}$$

Algoritmo 10

```
1  // Se tomarán en cuenta como operaciones la comparación arreglo a, asignaciones tanto de temp como de arreglo a
2
3  func OrdenamientoIntercambio(a,n){
4      for (int i = 0; i < n-1; ++i)              ->  n-1 (número de ejecuciones del ciclo)
5      {
6          for (int j = i; j < n; ++j)            ->  n-1-i (número de ejecuciones del ciclo)
7          {
8              if (a[j]<a[i])                     ->  1 (comparación)
9              {
10                 temp = a[i];                   ->  1 (asignación)
11                 a[i] = a[j];                   ->  1 (asignación)
12                 a[j] = temp;                   ->  1 (asignación)
13                 // Por este camino se tiene un total de 4 operaciones
14             }
15             // Si no entra al if, únicamente realiza una operación de comparación
16         }
17     }
18 }
19 fin
```

Mejor Caso

Si el arreglo se encuentra ordenando de manera ascendente, las instrucciones dentro del bloque if nunca se ejecutarán, por tanto, su fórmula sería la siguiente:

$$f_t(n) = \left(\frac{n}{2}\right)(n-1)(1) = \frac{n^2-n}{2}$$

Peor Caso

En caso de que el arreglo se encuentre de manera descendente, se ejecutaría siempre el bloque if, por tanto, la fórmula sería la siguiente:

$$f_t(n) = \left(\frac{n}{2}\right)(n-1)(4) = 2n^2 + 2n$$

Caso Medio

En este caso tenemos únicamente dos posibles caminos, uno donde se ejecuta todo el bloque if, y otro donde no lo hace. Suponiendo que ambos tengan la misma posibilidad de ejecutarse, la fórmula sería la siguiente:

$$f_t(n) = \left(\frac{n}{2}\right)(n-1)(1)\left(\frac{1}{2}\right) + \left(\frac{n}{2}\right)(n-1)(4)\left(\frac{1}{2}\right) = \frac{n^2-n+4n^2-4n}{4} = \frac{5n^2-5n}{4}$$

Algoritmo 11

```
1  // Se tomarán en cuenta como operaciones la operación de módulo a y b
2
3  fun MaximoComunDivisor(m,n){
4      a = max(n,m);
5      b = min(n,m);
6      residuo = 1;
7      while(residuo > 0){
8          residuo = a%b;          ->  1 (asignación de módulo)
9          a = b;
10         b = residuo;
11     }
12     MaximoComunDivisor = a;
13     return MaximoComunDivisor;
14 }
15 fin
```

Mejor Caso

El mejor caso se tiene cuando n o m es igual a 1, ya que éste será el MCD y el ciclo sólo se ejecutará una vez, por tanto, la fórmula estaría dada por:

$$f_t(n) = 1$$

Peor Caso

Éste viene dado por dos números subsecuentes en la serie de Fibonacci debido a que su módulo vendrá a ser el número anterior de la serie y así sucesivamente hasta llegar al MCD. Por tanto, su función sería:

$$f_t(n) = \lceil ln(n * m) \rceil$$

Algoritmo 12

```
1  // Se tomarán en cuenta como operaciones las comparaciones y asignaciones entre auxiliares y el arreglo
2
3  Procedimiento BurbujaOptimizada(A,n){
4      cambios = true;
5      i = 0;
6      while(i<n-1 && cambios){                        ->  n-1 (comparaciones)
7          cambios = false;
8          for (int j = 0; j < (n-1)-i; ++j)           ->  n-1-i (comparaciones)
9          {
10             if (A[i]<A[j])                          ->  1 (comparación)
11             {
12                 aux = A[j];                         ->  1 (asignación)
13                 A[j] = A[i];                        ->  1 (asignación)
14                 A[i] = aux;                         ->  1 (asignación)
15                 cambios = true;
16                 // Si entra al if, tendremos un total de 4 operaciones
17             }
18             // Si no se entra al if, tendremos únicamente 1 operación
19         }
20         i++;
21     }
22 }
```

Mejor Caso

Se presenta cuando el arreglo se encuentra ordenado ascendentemente, ya que la bandera de cambios nunca se cambiaría y sólo haría una vez el recorrido externo, considerando que el recorrido interno lo hará de manera completa, tenemos que la función queda de la siguiente manera:

$$f_t(n) = (n-1)(1) = n-1$$

Peor Caso

Cuando el arreglo se encuentra ordenado de manera descendente, ya que en ese caso siempre se ejecutará el bloque de instrucciones dentro del if, y su función temporal quedaría como se muestra a continuación:

$$f_t(n) = \left(\frac{n}{2}\right)(n-1)(4) \; = \; 2n^2 - 2n$$

Caso Medio

Únicamente se pueden presentar dos casos, que se ejecute el bloque de instrucciones dentro del if, o que no lo haga. Asumiendo que ambos tienen la misma probabilidad de suceder, la fórmula sería:

$$f_t(n) = \left(\frac{n}{2}\right)(n-1)(1)\left(\frac{1}{2}\right) + \left(\frac{n}{2}\right)(n-1)(4)\left(\frac{1}{2}\right) = \frac{n^2 - n + 4n^2 - 4n}{4} = \frac{5n^2 - 5n}{4}$$

Algoritmo 13

```
1  // Se tomarán en cuenta como operaciones las comparaciones y asignaciones entre auxiliares y el arreglo
2
3  Procedimiento BurbujaSimple(A,n){
4      for (int i = 0; i < n-1; ++i)              ->  n-1 (comparaciones)
5      {
6          for (int j = 0; j < (n-1)-i; ++j)      ->  n-1-i (comparaciones)
7          {
8              if (A[j]>A[j+1])                   ->  1 (comparación)
9              {
10                 aux = A[j];                    ->  1(asignación)
11                 A[j] = A[j+1];                 ->  1(asignación)
12                 A[j+1] = aux;                  ->  1(asignación)
13                 // Si se ejecuta este bloque de instrucciones, tenemos 4 operaciones
14             }
15             // En caso de omitirse, únicamente tenemos 1 operación
16         }
17     }
18 }
19 fin
```

Mejor Caso

Se presenta cuando el arreglo se encuentra ordenado ascendentemente, por lo tanto se ejecutarán ambos ciclos con una única instrucción dentro, tenemos que la función queda de la siguiente manera:

$$f_t(n) = \left(\frac{n}{2}\right)(n-1)(1) = \frac{n^2-n}{2}$$

Peor Caso

Cuando el arreglo se encuentra ordenado de manera descendente, en ese caso siempre se ejecutará el bloque de instrucciones dentro del if, y su función temporal quedaría como se muestra a continuación:

$$f_t(n) = \left(\frac{n}{2}\right)(n-1)(4) = 2n^2-2n$$

Caso Medio

Únicamente se pueden presentar dos casos, que se ejecute el bloque de instrucciones dentro del if, o que no lo haga. Asumiendo que ambos tienen la misma probabilidad de suceder, la fórmula sería:

$$f_t(n) = \left(\frac{n}{2}\right)(n-1)(1)\left(\frac{1}{2}\right) + \left(\frac{n}{2}\right)(n-1)(4)\left(\frac{1}{2}\right) = \frac{n^2-n+4n^2-4n}{4} = \frac{5n^2-5n}{4}$$

Algoritmo 14

```
1  // Se tomarán en cuenta como operaciones las comparaciones
2
3  Procedimiento Ordena(a,b,c){
4      if (a>b)                          ->  1 (comparación)
5      {
6          if (a>c)                      ->  1 (comparación)
7          {
8              if (b>c)                  ->  1 (comparación)
9              {
10                 salida(a,b,c);
11                 // En este camino tenemos un total de 3 operaciones
12             }
13             else
14                 salida(a,c,b);
15                 // En este camino tenemos un total de 3 operaciones
16         }
17         else
18             salida(c,a,b);
19             // En este camino tenemos un total de 2 operaciones
20     }
21     else{
22         if (b>c)
23         {
24             if (a>c)
25             {
26                 salida(b,a,c);
27                 // En este camino tenemos un total de 3 operaciones
28             }
29             else
30                 salida(b,c,a);
31                 // En este camino tenemos un total de 3 operaciones
32         }
33         else
34             salida(c,b,a);
35             // En este camino tenemos un total de 2 operaciones
36     }
37 }
38 fin
```

Mejor Caso

Hay dos posible mejores casos, cuando $c \leq a < b$ o $c \geq b \geq a$, para estos casos la función temporal sería:

$$f_t(n) = 2$$

Peor Caso

Tenemos cuatro posibles peores casos, estos son:

- $a > b, a > c \; y \; b > c$
- $a > b, a > c \; y \; b \leq c$

- $a \leq b, b > c \; y \; a > c$
- $a \leq b, b > c \; y \; a \leq c$

En estos casos, tenemos que la función temporal está dada por:

$$f_t(n) = 3$$

Caso Medio

Como es posible apreciarse, tenemos seis posibles caminos dentro del procedimientos, por tanto, asumiremos que tienen la misma probabilidad de suceder y eso nos daría la siguiente función temporal:

$$f_t(n) = (2)\left(\frac{1}{6}\right) + (2)\left(\frac{1}{6}\right) + (3)\left(\frac{1}{6}\right) + (3)\left(\frac{1}{6}\right) + (3)\left(\frac{1}{6}\right) + (3)\left(\frac{1}{6}\right) = \frac{2+2+3+3+3+3}{6} = \frac{8}{3}$$

Algoritmo 15

```
1  // Se tomarán en cuenta como operaciones las comparaciones y asignaciones
2  // Se asume que el rango va de 1 a n
3
4  func Seleccion(A,n)
5      Para k=0 hasta n-2 hacer                ->  n-1 (comparaciones)
6          p = k
7          Para i=k+1 hasta n-1 hacer          ->  n-1-k (comparaciones)
8              Si A[i]<A[p] entonces           ->  1 (comparación)
9                  p = i                       ->  1 (asignación)
10                 // Este camino nos deja 2 operaciones como resultado
11             Fin Si
12             // Si no se sigue el if, se tiene 1 operación solamente
13         Fin Para
14         temp = A[p]                         ->  1 (asignación)
15         A[p] = A[k]                         ->  1 (asignación)
16         A[k] = temp                         ->  1 (asignación)
17         // Estas 3 operaciones siempre se harán
18     Fin Para
19 Fin func
```

Mejor Caso

Cuando el arreglo se encuentra ordenado ascendentemente, nunca se seguirán las instrucciones del if, lo que nos daría como resultado la siguiente función:

$$f_t(n) = (n-1)(3 + \left(\frac{n}{2}\right)(1)) = (n-1)\left(\frac{6+n}{2}\right) = \frac{6n - 6 + n^2 - n}{2} = \frac{n^2 + 5n - 6}{2}$$

Peor Caso

Si se encontrase ordenado de manera descendente, siempre se estaría ejecutando la asignación del if, por tanto, resultaría la siguiente función:

$$f_t(n) = \ (n-1)(3+\left(\frac{n}{2}\right)(2)) \ = (n-1)(3+n) = 3n-3+n^2-n \ = n^2+2n-3$$

Caso Medio

En este caso tenemos dos posibles caminos, descritos arriba donde podemos o no ejecutar la asignación del bloque if. Asumiendo que tienen la misma posibilidad de ser ejecutados, resulta en un promedio de las dos funciones de arriba:

$$f_t(n) = \left(\frac{n^2+5n-6}{2}\right)\left(\frac{1}{2}\right) + (n^2+2n-3)\left(\frac{1}{2}\right) = \frac{n^2+5n-6+2n^2+10n-6}{4} = \frac{3n^2+15n-12}{4}$$