



Grado en Ingeniería Información

Estructura de Datos y Algoritmos

Sesión 3

Curso 2023-2024

Marta N. Gómez

- Criterios de valoración.
- ♦ Coste temporal.
- ♦ Coste asintótico.
- ♦ Cálculo del tamaño del problema.
- ♦ Análisis del Mejor y Peor caso.
- ♦ Notaciones asintóticas.
- ♦ Cálculo de la eficiencia.
- ♦ Coste espacial.



- **Criterios de valoración.**

- ♦ Coste temporal.
- ♦ Coste asintótico.
- ♦ Cálculo del tamaño del problema.
- ♦ Análisis del Mejor y Peor caso.
- ♦ Notaciones asintóticas.
- ♦ Cálculo de la eficiencia.
- ♦ Coste espacial.



¿Cómo se puede saber qué algoritmo es el mejor?

Existen diferentes criterios:

- Legibilidad
- Usabilidad/interfaz
- Facilidad de mantenimiento
- Velocidad de ejecución
- Necesidad de memoria
- Tiempo de desarrollo
- Elegancia
- etc.

¿Cómo se puede saber qué algoritmo es el mejor?

Existen diferentes criterios:

- Legibilidad
- Usabilidad/interfaz
- Facilidad de mantenimiento
- **Velocidad de ejecución**
- **Necesidad de memoria**
- Tiempo de desarrollo
- Elegancia
- etc.

Criterios de eficiencia:

- **Espacio**
- **Tiempo**



Complejidad Computacional



Principios básicos:

- Elegir las **estructuras de datos** adecuadas.
- Utilizar **algoritmos** eficientes para manejar las estructuras de datos elegidas.

Principios del Análisis de Algoritmos

- Independiente del **ordenador**.
- Independiente del **lenguaje de programación**.
- Independiente de **detalles de la implementación** (tipos de datos, sentencias, etc.).

Hay que analizar **algoritmos** y no programas

- Criterios de valoración.
- ♦ **Coste temporal.**
- ♦ Coste asintótico.
- ♦ Cálculo del tamaño del problema.
- ♦ Análisis del Mejor y Peor caso.
- ♦ Notaciones asintóticas.
- ♦ Cálculo de la eficiencia.
- ♦ Coste espacial.



```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int potencia = 10 * 10;
```

```
    cout << "\n\n\t10^2 = " << potencia;
```

```
    cout << "\n\n\t";
```

```
    return 0;
```

```
}
```

Forma 1: Producto

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int potencia = 10 * 10;
```

```
    cout << "\n\n\t10^2 = " << potencia;
```

```
    cout << "\n\n\t";
```

```
    return 0;
```

```
}
```

Forma 1: Producto

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int potencia{0};
```

```
    for (int i{0}; i < 10; i++)
```

```
    {
```

```
        potencia += 10;
```

```
    }
```

```
    cout << "\n\n\t10^2 = " << potencia;
```

```
    cout << "\n\n\t";
```

```
    return 0;
```

```
}
```

Forma 2: Suma

Calcular 10^2

Complejidad temporal

```
#include <iostream>

using namespace std;

int main() Forma 1: Producto
{
    int potencia = 10 * 10;

    cout << "\n\n\t10^2 = " << potencia;

    cout << "\n\n\t";
    return 0;
}
```

```
#include <iostream>

using namespace std;

int main() Forma 2: Suma
{
    int potencia{0};

    for (int i{0}; i < 10; i++)
    {
        potencia += 10;
    }

    cout << "\n\n\t10^2 = " << potencia;

    cout << "\n\n\t";
    return 0;
}
```

```
#include <iostream>

using namespace std;

int main() Forma 3: Incrementos
{
    int potencia{0};

    for (int i{0}; i < 10; i++)
    {
        for (int j{0}; j < 10; j++)
        {
            potencia ++;
        }
    }

    cout << "\n\n\t10^2 = " << potencia;

    cout << "\n\n\t";
    return 0;
}
```

El **tiempo** de cada programa **depende** de la **duración** de las **sentencias elementales utilizadas** (dependen del ordenador).

Programa	Productos	Sumas	Incrementos	Asignaciones	Comparaciones
forma1.cpp					
forma2.cpp					
forma3.cpp					

```
#include <iostream>

using namespace std;

int main()
{
    int potencia = 10 * 10;

    cout << "\n\n\t10^2 = " << potencia;

    cout << "\n\n\t";
    return 0;
}
```

Programa	Productos	Sumas	Incrementos	Asignaciones	Comparaciones
forma1.cpp	1			1	
forma2.cpp					
forma3.cpp					

```
#include <iostream>

using namespace std;

int main()
{
    int potencia{0};

    for (int i{0}; i < 10; i++)
    {
        potencia += 10;
    }

    cout << "\n\n\t10^2 = " << potencia;

    cout << "\n\n\t";
    return 0;
}
```

Programa	Productos	Sumas	Incrementos	Asignaciones	Comparaciones
forma1.cpp	1			1	
forma2.cpp		10	10	10+2 = 12	10+1=11
forma3.cpp					

```
#include <iostream>

using namespace std;

int main()
{
    int potencia{0};

    for (int i{0}; i < 10; i++)
    {
        for (int j{0}; j < 10; j++)
        {
            potencia ++;
        }
    }

    cout << "\n\n\t10^2 = " << potencia;

    cout << "\n\n\t";
    return 0;
}
```

Programa	Productos	Sumas	Incrementos	Asignaciones	Comparaciones
forma1.cpp	1			1	
forma2.cpp		10	10	12	11
forma3.cpp			$2(10 \cdot 10) + 10 = 210$	$10 + 2 = 12$	$(10 + 1) \cdot (10 + 1) = 11 \cdot 11 = 121$

Calcular n^2

Complejidad temporal

```
#include <iostream>
```

```
using namespace std;
```

```
int main()  
{
```

```
    int potencia, n;
```

```
    cout << "\n\n\tIndique un numero entero: ";
```

```
    cin >> n;
```

```
    potencia = n * n;
```

```
    cout << "\n\n\tPotencia = " << potencia;
```

```
    cout << "\n\n\t";
```

```
    return 0;
```

```
}
```

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int potencia{0}, n;
```

```
    cout << "\n\n\tIndique un numero entero: ";
```

```
    cin >> n;
```

```
    for (int i{0}; i < n; i++) {
```

```
        potencia += n;
```

```
    }
```

```
    cout << "\n\n\tPotencia = " << potencia;
```

```
    cout << "\n\n\t";
```

```
    return 0;
```

```
}
```

Forma 1: Producto

Forma 2: Suma

Forma 3: Incrementos

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int potencia{0}, n;
```

```
    cout << "\n\n\tIndique un numero entero: ";
```

```
    cin >> n;
```

```
    for (int i{0}; i < n; i++)
```

```
    {
```

```
        for (int j{0}; j < n; j++) {
```

```
            potencia ++;
```

```
        }
```

```
    }
```

```
    cout << "\n\n\tPotencia = " << potencia;
```

```
    cout << "\n\n\t";
```

```
    return 0;
```

```
}
```

El **tiempo** de alguno de los programas **depende** del **valor de n** :

- Cuanto mayor es n , más cuesta resolver el problema.
- El **tamaño** del problema es n .
- Hay que determinar el **coste del algoritmo** en función de n .
- Por tanto, se puede representar gráficamente el tamaño temporal de los algoritmos en función del tamaño de n .

Programa	Productos	Sumas	Incrementos	Asignaciones	Comparaciones
forma1.cpp					
forma2.cpp					
forma3.cpp					

```
#include <iostream>

using namespace std;

int main()
{
    int potencia, n;

    cout << "\n\n\tIndique un numero entero: ";
    cin >> n;
    potencia = n * n;
    cout << "\n\n\tPotencia = " << potencia;

    cout << "\n\n\t";
    return 0;
}
```

Programa	Productos	Sumas	Incrementos	Asignaciones	Comparaciones
forma1.cpp	1			1	
forma2.cpp					
forma3.cpp					

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int potencia{0}, n;
```

```
    cout << "\n\n\tIndique un numero entero: ";
```

```
    cin >> n;
```

```
    for (int i{0}; i < n; i++) {
```

```
        potencia += n;
```

```
    }
```

```
    cout << "\n\n\tPotencia = " << potencia;
```

```
    cout << "\n\n\t";
```

```
    return 0;
```

```
}
```

Programa	Productos	Sumas	Incrementos	Asignaciones	Comparaciones
forma1.cpp	1			1	
forma2.cpp		n	n	n + 2	n + 1
forma3.cpp					

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
{
    int potencia{0}, n;

    cout << "\n\n\tIndique un numero entero: ";
    cin >> n;
    for (int i{0}; i < n; i++)
    {
        for (int j{0}; j < n; j++) {
            potencia ++;
        }
    }
    cout << "\n\n\tPotencia = " << potencia;

    cout << "\n\n\t";
    return 0;
}
```

Programa	Productos	Sumas	Incrementos	Asignaciones	Comparaciones
forma1.cpp	1			1	
forma2.cpp		n	n	n + 2	n + 1
forma3.cpp			$(n*n)+(n*n)+n=2n^2 + n$	n + 2	$(n+1)*(n+1)=n^2 + 2n + 1$