Ciencia y Tecnología
Secretaría de Ciencia, Humanidades, Tecnología e Innovación

CIMAT
Centro de Investigación en Matemáticas, A.C.

Algoritmos y optimización de código

Unidad 1. Introducción a la complejidad

Introducción a la complejidad

- Considere el problema de ordenar una lista $L = \{a_1, a_2, \ldots, a_n\}$ números dados en un orden arbitrario.
  - Ordenamiento por insersión
  - Ordenamiento por unión

Notación asintótica

- Notación O

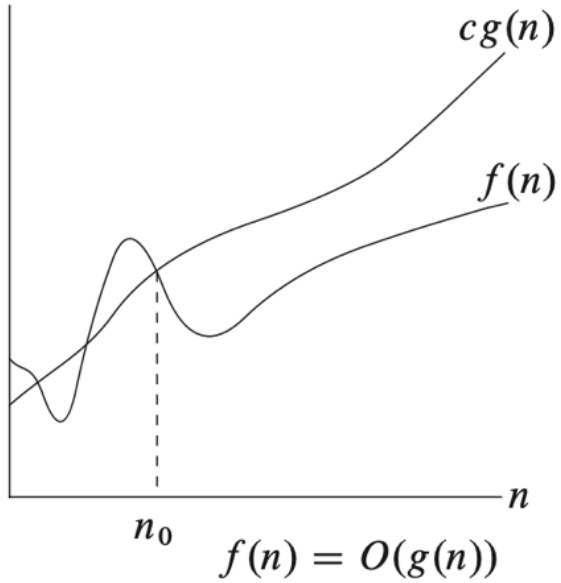
$O\big(g(n)\big) = \{f(n)$: si existen constantes positivas c y n_0 tales que el tiempo de ejecución $0 \leq f(n) \leq cg(n)$ para toda $n \geq n_0\}$

Notación asintótica

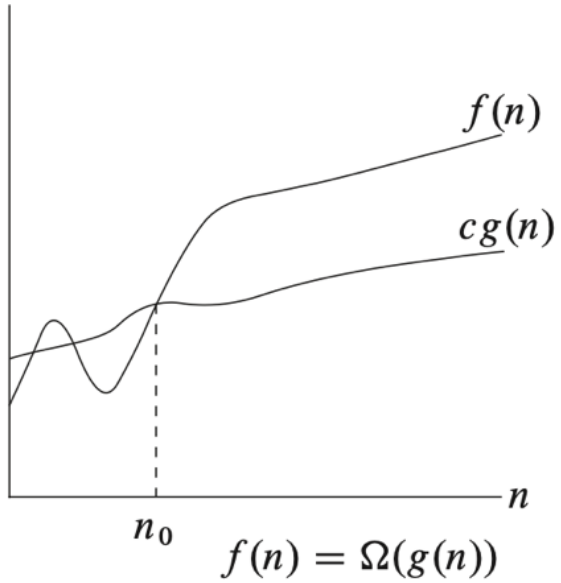
- Notación Ω

$\Omega\big(g(n)\big) = \{f(n)$: si existen constantes positivas c y n_0 tales que el tiempo de ejecución $0 \leq cg(n) \leq f(n)$ para toda $n \geq n_0\}$

Notación asintótica



$f(n) = O(g(n))$



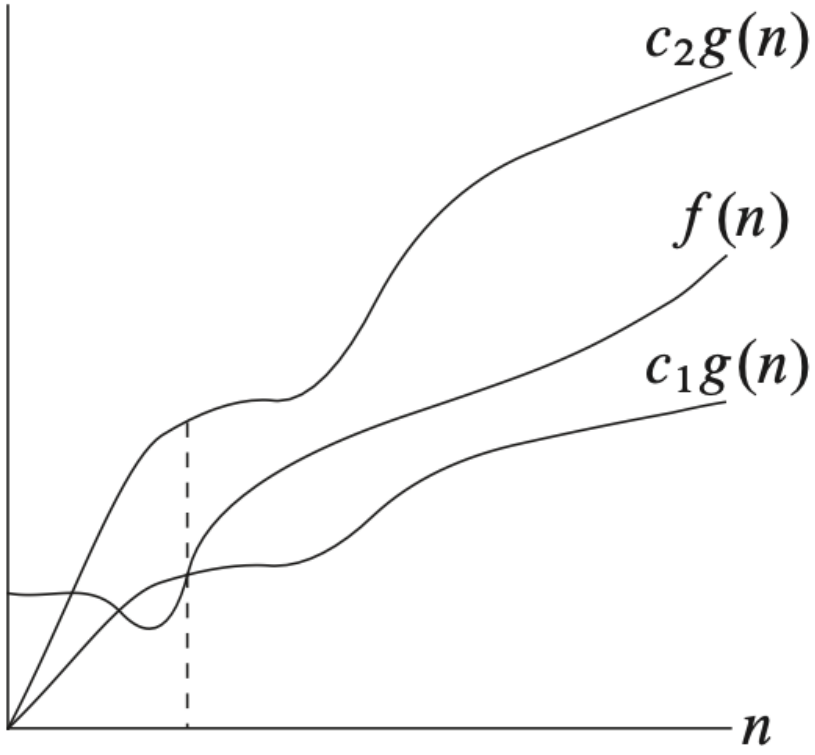
$f(n) = \Omega(g(n))$

Notación asintótica

- Notación Θ

$\Theta\big(g(n)\big) = \{f(n)$: si existen constantes positivas c_1, c_2 y n_0 tales que el tiempo de ejecución $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ para toda $n \geq n_0\}$

Notación asintótica



$$f(n) = \Theta(g(n))$$

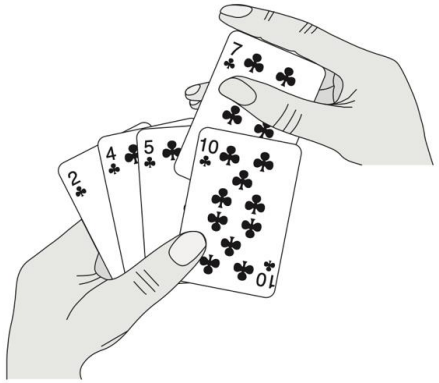
Ejercicios

Ejercicios.

1. Demostrar que $2n^2 \notin O(n)$
2. Demostrar que $f(n) = \frac{1}{2}n^2 - 3n$ pertenece al $\Theta(n^2)$
3. Demostrar que $f(n) = 7n^3$ no pertenece a $\Theta(n^2)$

Algoritmos de ordenamiento

- Un algoritmo de ordenamiento consiste en recibir una secuencia de n números $\langle a_1, a_2, \dots, a_n \rangle$ y proporcionar como salida una permutación de dicha secuencia $\langle a'_1, a'_2, \dots, a'_n \rangle$ de forma tal que $a'_1 \leq a'_2 \leq \cdots \leq a'_n$. Así por ejemplo, si recibimos como entrada la secuencia de números $\langle 45, 23, 12, 58, 89, 61 \rangle$, un algoritmo de ordenamiento debería ser capaz de dar como salida la siguiente permutación de la entrada del algoritmo $\langle 12, 23, 45, 58, 61, 89 \rangle$.



Ordenamiento por insersión

- El Seudocódigo 1 presenta el algoritmo de ordenamiento por inserción

```
ORDENAMIENTO-INSERCION (A)
1. for i=1 to A.lenght
2.       key= A[j]
3.       // Inserta el elemento A[j] en su
posición correcta en el arreglo A[1..j-1]
4.       i=j-1
5.       while i>0 and A[i]>key
6.            A[i+1]= A[i]
7.            i=i-1
8.       A[i+1]= key
```

Seudocódigo 1. Algoritmo de ordenamiento por inserción

Ordenamiento por insersión

- ¿Cómo calculamos el tiempo de ejecución de este algoritmo?

Algoritmo de ordenamiento por Merge

- **Divide** el problema en problemas más pequeños que la instancia original
- **Conquista** los subproblemas resolviendo instancias más pequeñas recursivamente. Si el tamaño de la instancia es suficientemente pequeña, resuelve de forma exhaustiva.
- **Combina** las soluciones de los subproblemas en la solución del problema original.

Algoritmo de ordenamiento por Merge

- Ordenamiento Merge

```
ORDENAMIENTO-MERGE (A,p,q)
1. if p<q
2.     q=⌊(p + q)/2⌋
3.     ORDENAMIENTO-MERGE(A,p,q)
4.     ORDENAMIENTO-MERGE(A,q+1,r)
5.     UNE(A,p,q,r)
```

Seudocódigo 2. Algoritmo de ordenamiento por MERGE

Algoritmo de ordenamiento por Merge

- Ordenamiento Merge

```
UNE (A,p,q,r)
1. n_1 = q − p + 1
2. n_2 = r − q
3. Sean L[1..n_1 + 1]  y R[1..n_2 + 1] los nuevos arreglos a procesar
4. for i = 1 to n_1
5.       L[i] = A[p + i − 1]
6. for j = 1 to n_2
7.       R[j] = A[q + j]
8. L[n_1 + 1] = ∞
9. R[n_2 + 1] = ∞
10. i = 1
11. j = 1
12. for k = p to r
13.       if L[i] ≤ R[j]
14.             A[k] = L[i]
15.             i = i + 1
16.       else A[k] = R[j]
17.             j = j + 1
```

Seudocódigo 3. Algoritmo UNE

Ordenamiento por Merge

- ¿Cómo calculamos el tiempo de ejecución de este algoritmo?

Algoritmos de ordenamiento

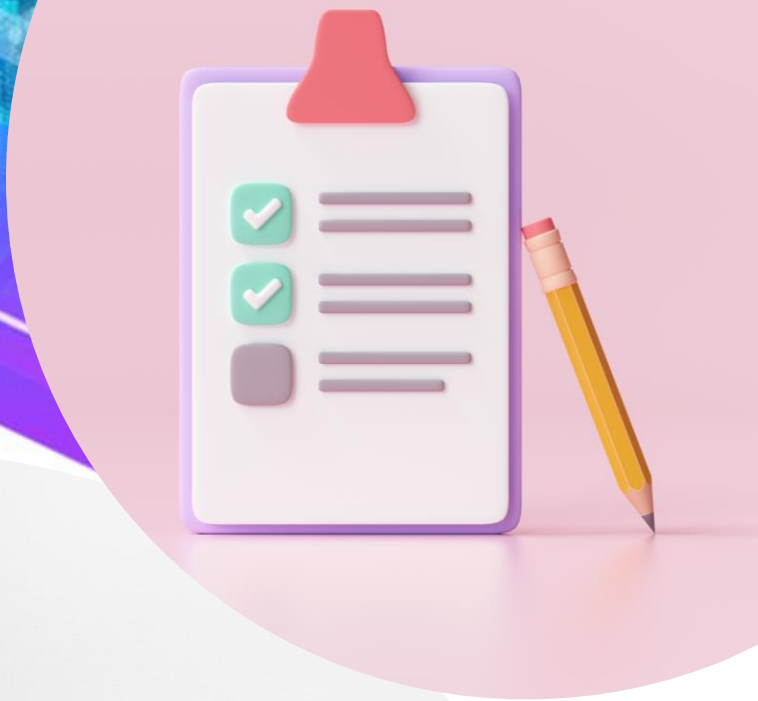
Ejercicios.

1. Generar un total de 10 instancias tales que cada una de éstas incluya 1000000 elementos de tipo entero distribuidos normalmente en valores del 1 al 1000000.
2. Implementar el algoritmo de ORDENAMIENTO-INSERCION (A) para cada una de estas instancias
3. Implementar el algoritmo de ORDENAMIENTO-MERGE (A,p,q) para cada una de estas instancias
4. Realizar una tabla comparativa que muestre el tiempo de ejecución de dichos algoritmos. Indicar las características del equipo donde se ejecutó.

Ciencia y Tecnología
Secretaría de Ciencia, Humanidades, Tecnología e Innovación

CIMAT
Centro de Investigación en Matemáticas, A.C.

Algoritmos para problemas de búsqueda



Búsqueda líneal

```
BUSCA (A,v)
1. bandera=false
2. for i=1 to n
2.    if A[i]=v then
3.       Imprime "Elemento encontrado en la posición i"
4.       bandera=true
5. if (bandera=false) then
6.       Imprime "Elemento no encontrado"
```

Seudocódigo 4. Algoritmo de ordenamiento por MERGE

Búsqueda lineal

- Orden de ejecución

$$t(n) = O(n).$$

Búsqueda binaria

Ejercicios

- Implementar los algoritmos de búsqueda líneal y búsqueda binaria
- Implemente instancias para el caso de búsqueda lineal y el caso de búsqueda binaria
- Suponga que no tienes conocimiento acerca de un arreglo de tamaño n. Qué será más eficiente: utilizar búsqueda lineal o ordenar el arreglo y utilizar búsqueda binaria. Justifique su respuesta.