Introducción a las Menciones II: Computación Científica - Diseño de algorítmos

Graeme Candlish

graeme.candlish@ifa.uv.cl

Contenido

Algorítmos y pseudocódigo

Contenido

Algorítmos y pseudocódigo

Descripción de un algorítmo

Un programa es una implementación de un algorítmo. Podemos definir un algorítmo en un lenguaje natural (lenguaje humano).

Ejemplo: haciendo un té

1. Poner agua en el hervidor.
2. Hervir el agua.
3. Poner un bolsillo de té en una taza.
4. Poner el agua en la taza.
5. Si la persona quiere leche, poner leche en la taza.
6. Si la persona quiere azucar, poner azucar en la taza.
7. Si la persona quiere endulzante, ponerlo.
8. Remover el té.

La idea del pseudocódigo

Pseudocódigo:
Definición de un algorítmo en una forma muy cerca a la forma de un programa computacional.

El concepto de *pseudocódigo* no es muy preciso - depende del nivel de detalle que queremos usar.

Ejemplo: haciendo un té (pseudocódigo)

1. Hervir el agua.
2. En taza poner bolsillo y agua.
3. Si (quiere leche): en taza poner leche.
4. Si (quiere azucar): en taza poner azucar.
5. Si (no quiere azucar y quiere endulzante): en taza poner endulzante.
6. Remover el té.

Ejemplo: ordenar una lista de números

1. Elegir el primer par de números a, b de la lista.
2. Si (a $>$ b): intercambiar los dos números en la lista.
3. Si (b $>$ a): dejar los números donde están.
4. Cambiar al próximo par y aplicar las mismas operaciones.
5. Cuando hemos procesado toda la lista, repetir de nuevo, hasta que la lista esté ordenada (no hay intercambios necesarios).

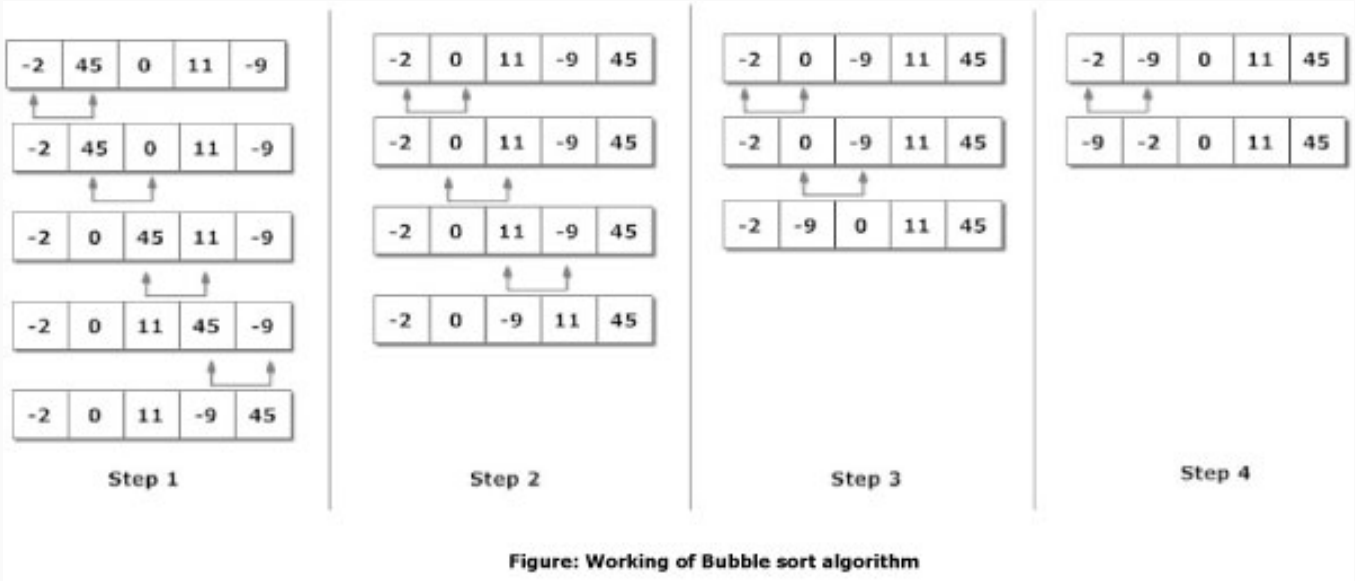


Figure: Working of Bubble sort algorithm

Ejemplo: ordenar una lista de números

Pseudocódigo (muy cerca a Python)

```
cambios = 0
while True:
    for par in lista:
        a, b = par
        if (a > b):
            lista[0] = b; lista[1] = a
            cambios += 1
    if (cambios == 0):
        break
```

Tarea

Escribir (en pseudocódigo) un algorítmo para hacer un sándwich de jamón y queso.

Tarea

Poner las siguientes partes en el órden correcto para un algorítmo de una simulación de N-cuerpos:

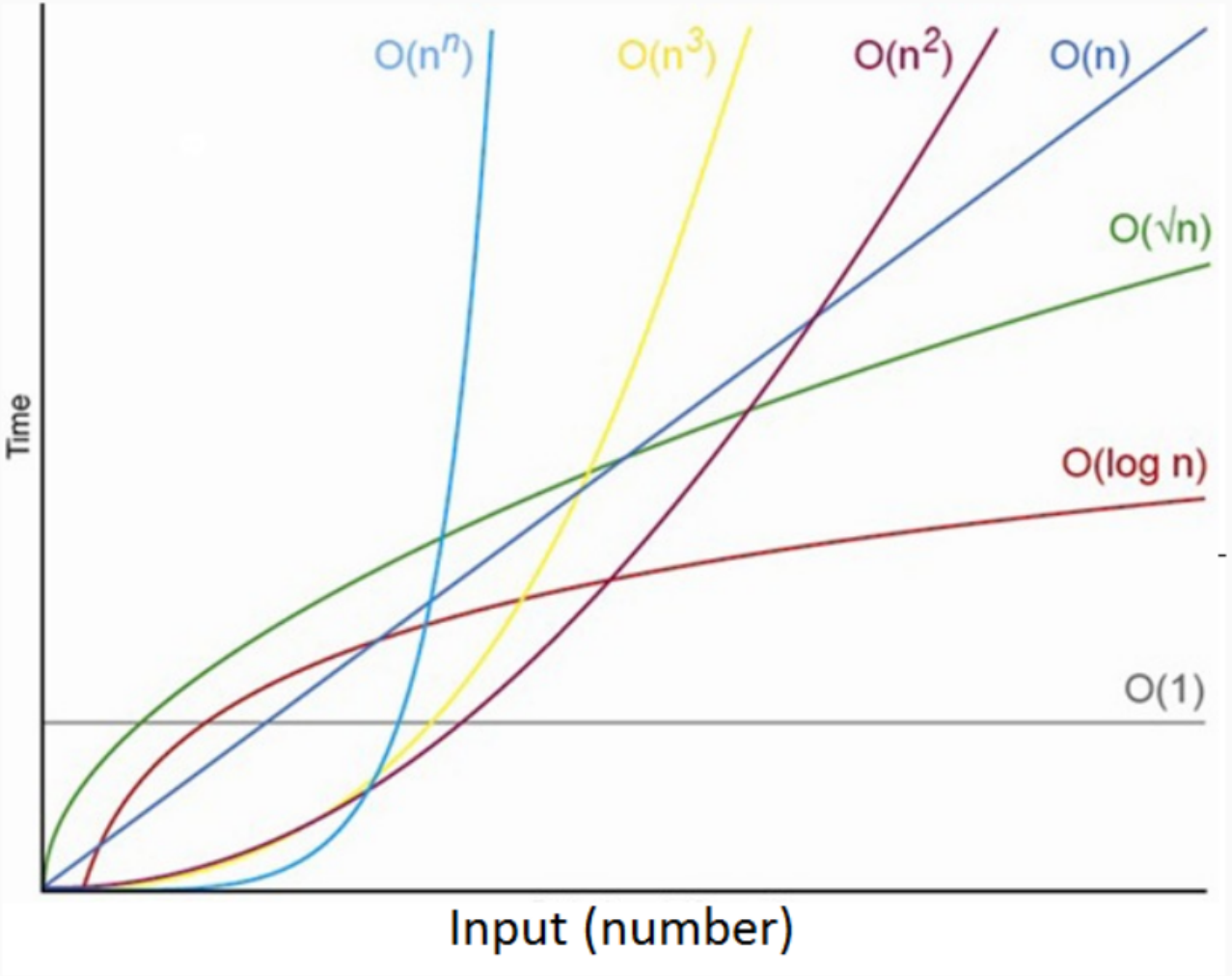
1. Repetir los pasos para todos los momentos del tiempo.
2. Calcular la velocidad de cada partícula de su aceleración.
3. Calcular la aceleración de cada partícula utilizando las fuerzas gravitacionales.
4. Mover las partículas según sus nuevas velocidades.
5. Calcular la fuerza gravitacional entre las partículas.

Complejidad algorítmica

Notación de $\mathcal{O}$ grande ("big-O"):

- Supongamos que tenemos un algoritmo que procesa N entradas.
- Si el algoritmo requiere αN pasos para terminar, donde α es un factor constante, el algoritmo tiene **complejidad algorítmica** de $\mathcal{O}(N)$.
- Si requiere $\alpha N^2 + N$ pasos, tiene complejidad algorítmica de $\mathcal{O}(N^2)$.

Complejidad algorítmica



Tarea

En el caso del algoritmo para ordenar números que vimos, determinar cuántos pasos necesitamos para ordenar las siguientes listas: Lista A = 3,2,1. Lista B = 4,3,2,1.

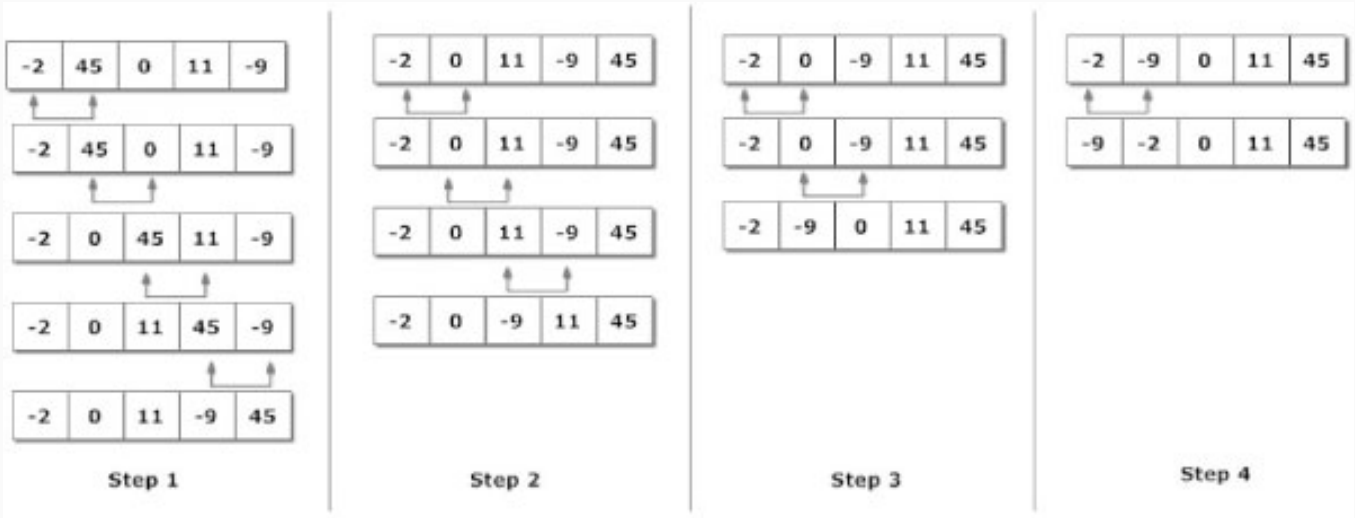


Figure: Working of Bubble sort algorithm

Complejidad algorítmica

- En la computación científica, necesitamos procesar muchos datos.
- Por lo tanto, requerimos algoritmos **eficientes**.
- En el caso de una simulación de N-cuerpos, calcular todas las fuerzas entre cada par de N partículas tiene complejidad $\mathcal{O}(N^2)$.
- En una simulación cosmológica hay millones de partículas... necesitamos otro método.