# Estructuras de Datos #5

## Algoritmos de Ordenamiento

# Ordenando en Bases de Datos

- Muchas posibilidades
  - Nombres en orden alfabético
  - Estudiantes por nota
  - Clientes por código postal
  - Venta de casas por precio
  - Ciudades por población
  - Países por PGB
  - Estrellas por magnitud

# Ordenamiento y Búsqueda

- Vimos que los arreglos pueden trabajar en conjunto para mejorar la velocidad

- Pero esto no tiene sentido si el ordenamiento se tarda una eternidad!
  - Por esto y por su utilidad, los algoritmos de ordenamiento han sido ampliamente investigados

| Estables | | | | |
|---|---|---|---|---|
| Nombre traducido | Nombre original | Complejidad | Memoria | Método |
| Ordenamiento de burbuja | Bubblesort | $O(n^2)$ | $O(1)$ | Intercambio |
| Ordenamiento de burbuja bidireccional | Cocktail sort | $O(n^2)$ | $O(1)$ | Intercambio |
| Ordenamiento por selección | Selection Sort | $O(n^2)$ | $O(1)$ | Intercambio |
| Ordenamiento por inserción | Insertion sort | $O(n^2)$ | $O(1)$ | Inserción |
| Ordenamiento por casilleros | Bucket sort | $O(n)$ | $O(n)$ | No comparativo |
| Ordenamiento por cuentas | Counting sort | $O(n+k)$ | $O(n+k)$ | No comparativo |
| Ordenamiento por mezcla | Merge sort | $O(n \log n)$ | $O(n)$ | Mezcla |
| Ordenamiento con árbol binario | Binary tree sort | $O(n \log n)$ | $O(n)$ | Inserción |
| | Pigeonhole sort | $O(n+k)$ | $O(k)$ | |
| Ordenamiento Radix | Radix sort | $O(nk)$ | $O(n)$ | No comparativo |
| | Distribution sort | $O(n^3)$ versión recursiva | $O(n^2)$ | |
| | Gnome sort | $O(n^2)$ | $O(1)$ | |

| Inestables | | | | |
|---|---|---|---|---|
| Nombre traducido | Nombre original | Complejidad | Memoria | Método |
| Ordenamiento Shell | Shell sort | $O(n^{1.25})$ | O(1) | Inserción |
| | Comb sort | $O(n \log n)$ | O(1) | Intercambio |
| Ordenamiento por selección | Selection sort | $O(n^2)$ | O(1) | Selección |
| Ordenamiento por montículos | Heapsort | $O(n \log n)$ | O(1) | Selección |
| | Smoothsort | $O(n \log n)$ | O(1) | Selección |
| Ordenamiento rápido | Quicksort | Promedio: $O(n \log n)$,<br>peor caso: $O(n^2)$ | O(log *n*) | Partición |
| | Several Unique Sort | Promedio: $O(n\ u)$,<br>peor caso: $O(n^2)$;<br>u=n; u = número único de registros | | |
| **Cuestionables, imprácticos** | | | | |
| Nombre traducido | Nombre original | Complejidad | Memoria | Método |
| | Bogosort | $O(n \times n!)$, peor: no termina | | |
| | Pancake sorting | $O(n)$, excepto en<br>máquinas de Von Neumann | | |
| Ordenamiento Aleatorio | Randomsort | Promedio: O(n!) Peor: No termina | | |

# Algoritmos Básicos de Ordenamiento

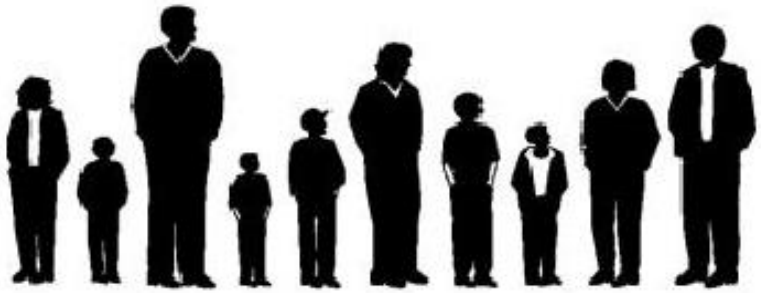
- Burbuja
- Selección
- Inserción

- Aunque estos son:
  - Más sencillos
  - Más lentos

- Algunas veces son mejores que los avanzados
- Y algunas veces los métodos avanzados están construidos basados en ellos

# Ejemplo

- Desordenado:



- Ordenado:

- Por supuesto, un computador no tiene el lujo que tenemos nosotros…

# Ordenamientos Simples

- Los tres algoritmos implican dos pasos básicos, que se ejecutan repetidamente hasta que se ordenan los datos
  - Comparar dos ítems
  - intercambiar dos elementos, o copiar un elemento

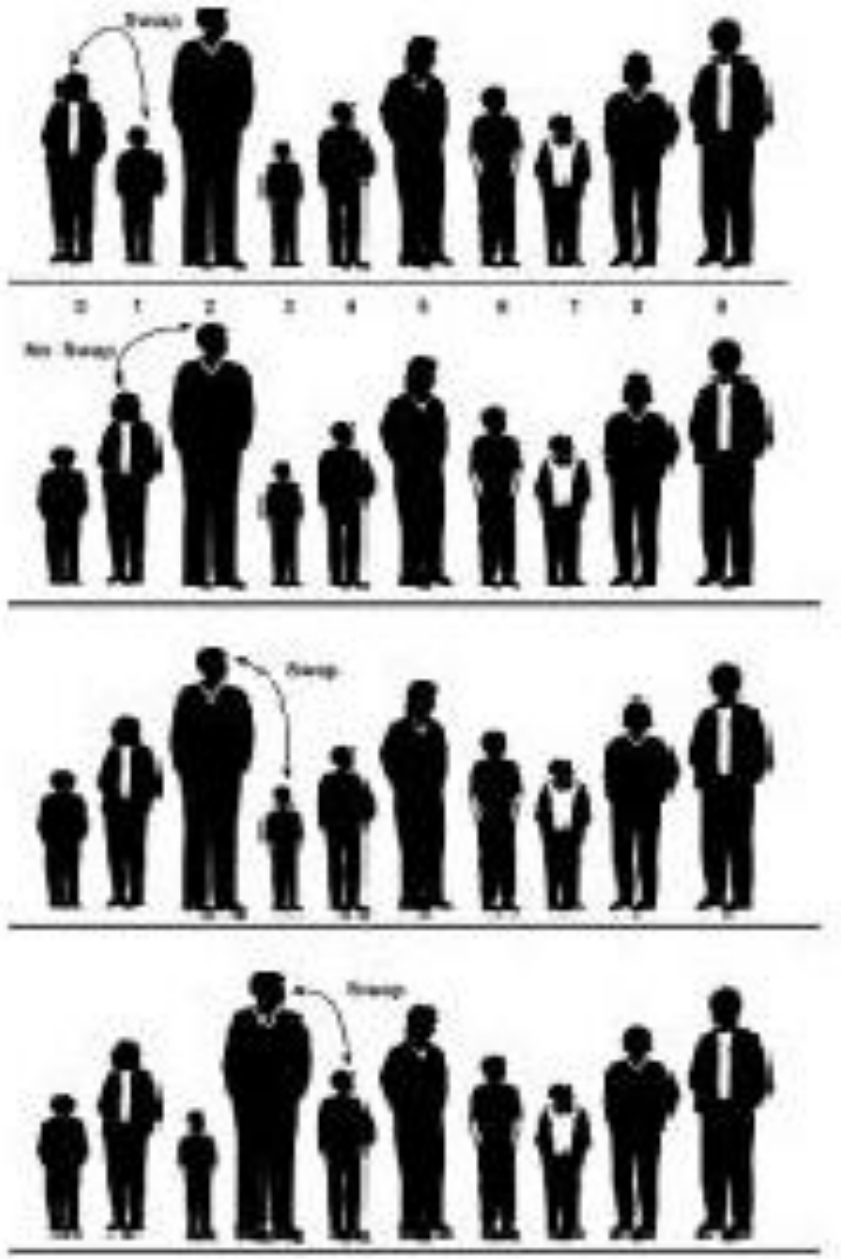
- Se diferencian en los detalles y el orden de las operaciones

# Ordenar#1: Burbuja

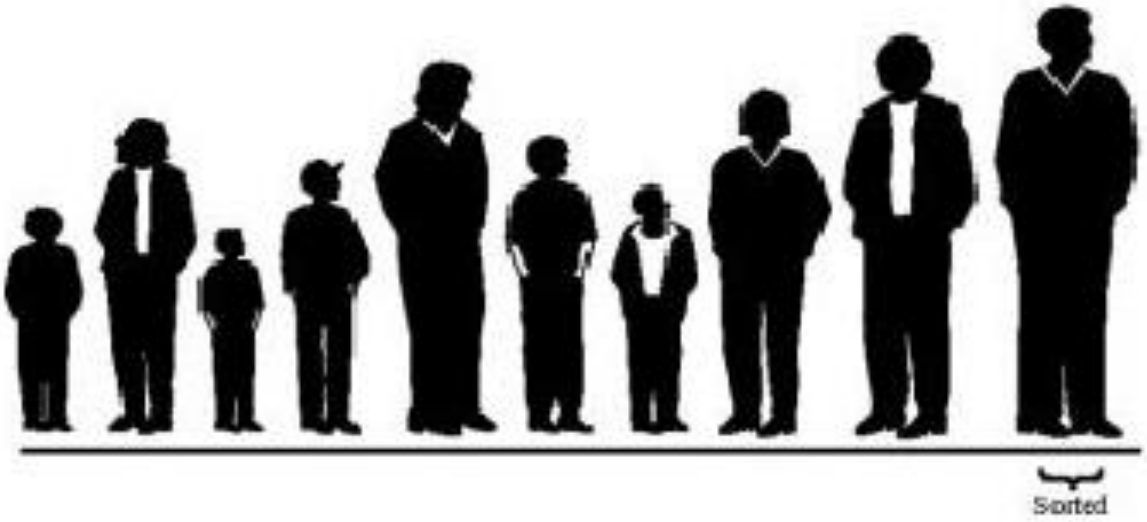
- Camino a seguir:
  - Suponga que usted es "miope"
  - Sólo  puede ver a dos jugadores adyacentes al mismo tiempo
  - ¿Cómo ordenar?

- Así es como funciona burbuja!
  - 1. Comience en la posición i = 0
  - 2. Compare posición i y i+1
  - 3. Si el jugador en la posición i es más alto que el de la i+1, intercambio (intercambio)
  - 4. Mueva una posición a la derecha

# Burbuja: Primer Paso



# Burbuja: Fin del primer paso




- Note: Ahora la persona más alta esta al final
  - ¿Por que?
- ¿Entendemos ahora el porque se llama este ordenamiento burbuja?

# Conteo de Operaciones

- Primera Pasada, para un arreglo de tamaño n:
  - ¿Cuántas comparaciones se han hecho?
  - ¿Cuántos intercambios (pero caso) se han hecho?

- Ahora nuevamente debemos partir en cero y hacer lo mismo
  - Comparar
  - Intercambiar si se requiere
  - Moverse a la derecha
  - Pero esta vez, debemos de hacer un paso menos (¿Por que?)

- Sigue realizando estos pasos hasta que todos los jugadores estén en orden

# Veamos un ejemplo...

- Use ordenamiento por burbuja para ordenar un arreglo de 10 enteros:

- 30 45 8 204 165 95 28 180 110 40

# ¿Cuantas operaciones se realizan en total?

- Primera pasada: n-1 comparaciones, n-1 intercambios
- Segunda pasada: n-2 comparaciones, n-2 intercambios
- Tercera pasada: n-3 comparaciones, n-3 intercambios
- (n-1)-m pasada: 1 comparación, 1 intercambio
- Finalmente está el arreglo ordenado

- Tenemos (peor caso):
  - $(n-1)+(n-2)+\ldots+1$ comparaciones $= n(n-1)/2 = 0.5(n^2-n)$
  - $(n-1)+(n-2)+\ldots+1$ intercambios $= n(n-1)/2 = 0.5(n^2-n)$

- Total: $2*(0.5(n^2-n)) = n^2-n = O(n^2)$

# Ordenar #2: Selección

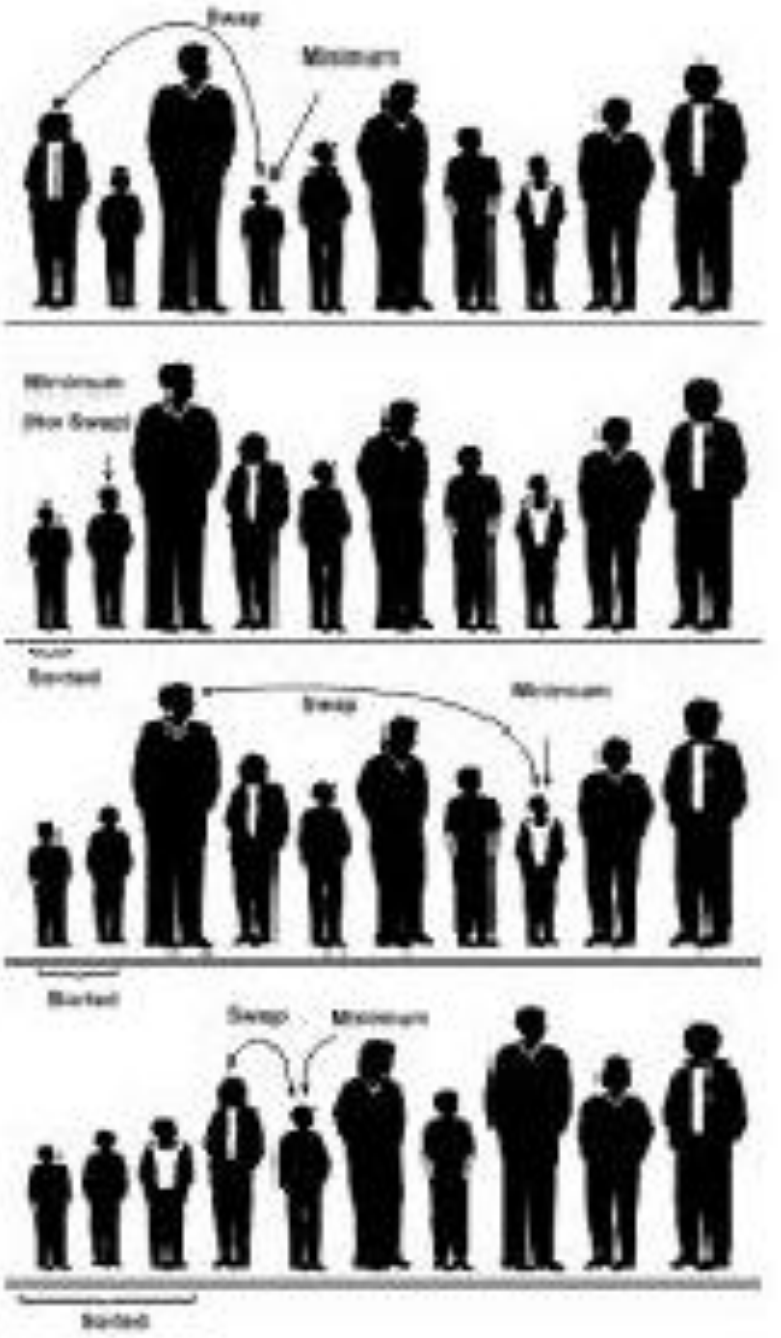
- Propósito:
  - Mejorar la velocidad de burbuja
  - Número de comparaciones: $O(n^2)$
  - Número de intercambios: $O(n)$

- Volvamos al equipo de baseball…
  - Ahora, ya no comparamos los jugadores sólo con su vecino
  - Más bien, debemos "recordar" las alturas

# De que se trata

- Haz un paso a través de todos los jugadores
  - Encuentra el más pequeño
- Intercambia con aquel a la izquierda de la línea
  - En la posición 0
- Ahora, el más a la izquierda se ordena
- Encuentre el más pequeño entre los (n-1) jugadores restantes
- Intercambie aquel con el jugador de la posición 1
- Y así sucesivamente y así sucesivamente …

# Selección en Acción



# Conteo de Operaciones

- Primer paso, para un arreglo de tamaño n:
  - ¿Cuántas comparaciones se hacen?
  - ¿Cuántos (peor caso) intercambios se hace?

- Ahora tenemos que comenzar de nuevo en la posición uno y hacemos lo mismo
  - Encontrar el más pequeño
  - Intercambiar con la posición uno

- Siga haciendo esto hasta que todos los jugadores están en orden

# Veamos un ejemplo ...

- Use selección para ordenar un arreglo de 10 números enteros:

- 30 45 8 204 165 95 28 180 110 40

# ¿Cuántas operaciones se realizan en total?

- Primera pasada: n-1 comparaciones, 1 intercambio
- Segunda pasada: n-2 comparaciones, 1 intercambio
- Tercera pasada: n-3 comparaciones, 1 intercambio
- (n-1)-m pasada: 1 comparaciones, 1 intercambio
- Finalmente hemos ordenado el arreglo

- Tenemos (peor caso):
  - $(n-1)+(n-2)+\ldots+1$ comparaciones $= n(n-1)/2 = 0.5(n^2 - n)$
  - $1+1+\ldots+1$ intercambios $= n-1$

- Total: $0.5(n^2 - n) + (n - 1) = 0.5n^2 + 0.5n - 1 = O(n^2)$

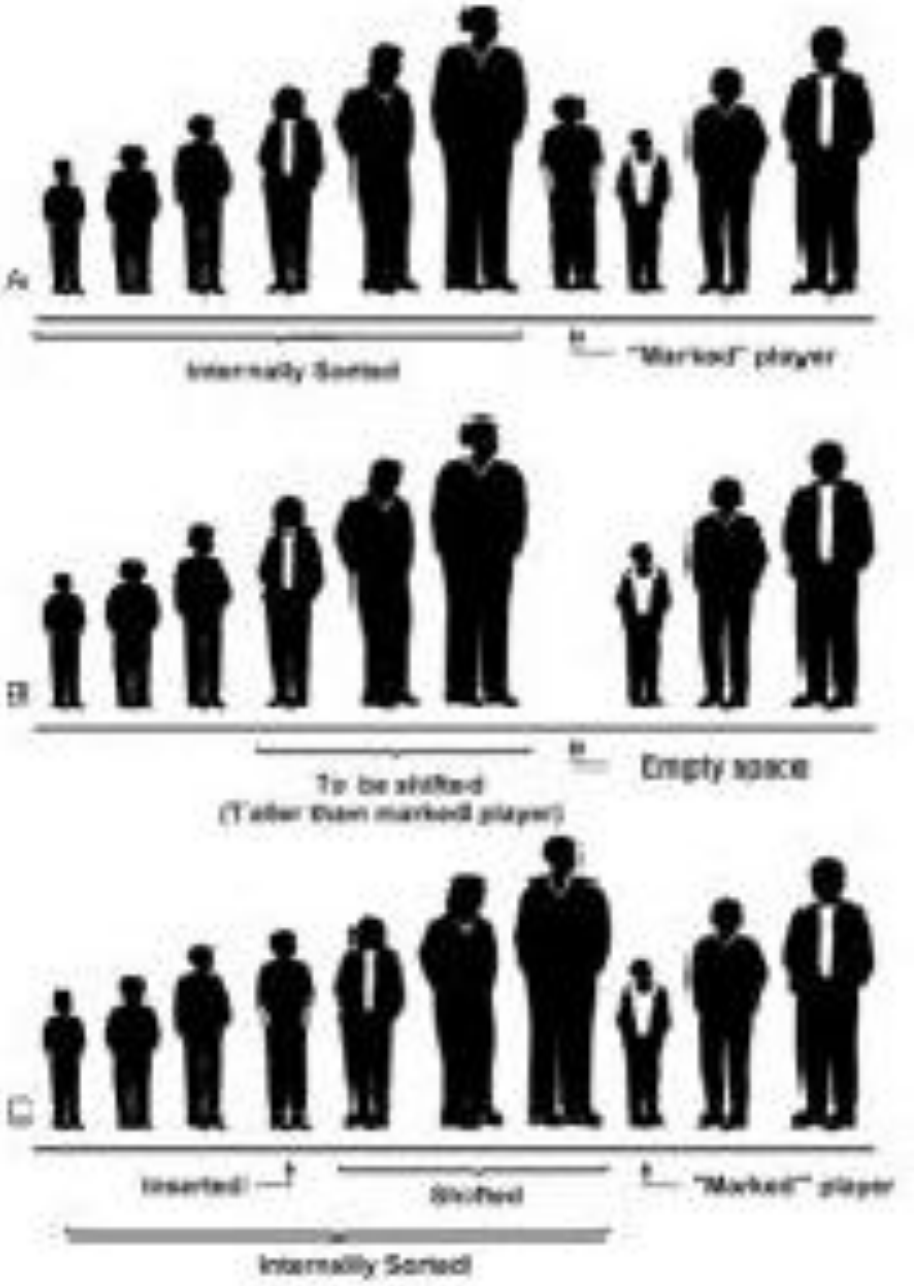
# Ordenar #3: Inserción

- En la mayoría de los casos, el mejor …
  - 2x más rápido que la ordenación de burbuja
  - Algo más rápido que selección en la mayoría de los casos

- Un poco más complejo que los otros dos
- Algoritmos más avanzados (quicksort) lo utilizan como una etapa

# Proceder..

- El sub-arreglo de la izquierda está “parcialmente ordenado”
- Comience con el primer elemento
- El jugador inmediatamente a la derecha es "marcado".
- El jugador "marcado" se inserta en el lugar correcto en el arreglo parcialmente ordenado
  - Remueva primero
  - El jugador marcado 'camina' a la izquierda
  - Desplazar elementos apropiados hasta que llegamos a uno más pequeño



# Veamos un ejemplo...

- Use inserción para ordenar 10 números enteros:

- 30 45 8 204 165 95 28 180 110 40

# Conteo de Operaciones

- Primera Pasada, para un arreglo de tamaño n:
  - ¿Cuántas comparaciones fueron hechas?
  - ¿Cuantos intercambios fueron hechos?
    - ¿Hubo alguna? ¿Qué había?

- Ahora tenemos que empezar de nuevo en la posición dos y hacer lo mismo
  - Mueva el jugador marcado al punto correcto

- Siga haciendo esto hasta que todos los jugadores estén en orden

# ¿Cuántas Comparaciones se hicieron en total?

- Primera pasada: 1 comparación, 1 copia
- Segunda pasada: 2 comparaciones, 2 copias
- Tercera pasada: 3 comparaciones, 3 copias
- (n-1)-ma pasada: n-1 comparaciones, n-1 copias
- Finalmente el arreglo esta ordenado

- Tenemos (peor caso):
  - $(n-1)+(n-2)+\ldots+1$ comparaciones $= n(n-1)/2 = 0.5(n^2 - n)$
  - $(n-1)+(n-2)+\ldots+1$ copias $= n(n-1)/2 = 0.5(n^2 - n)$
- Total: $2*(0.5(n^2+n)) = n^2 - n = O(n^2)$

# ¿Por qué estamos afirmando que es mejor que la ordenación por selección?

- Un intercambio es una operación cara. Una copia no lo es.
  - Para ver esto, ¿Cuántas copias son requeridas por intercambio?
  - Selección/burbuja usan intercambios, inserción usa copias.

- En segundo lugar, si un arreglo está ordenado o casi ordenado, la inserción se ejecuta en aproximadamente O (n) tiempo. ¿Por qué?
  - ¿Cuántas comparaciones necesitarías por iteración? ¿Y cuántas copias?
  - ¡Burbuja y selección requerirían O ($n^2$) comparaciones aun cuando el arreglo estuviese inicialmente ordenado!

# Tarea: Implementación

- Implementar los 3 métodos de ordenamiento en Java
- Usar vectores

# Comparación Ordenamiento: Resumen

- Burbuja– apenas se usa
  - Muy lento, a menos que el número de datos sea muy pequeño
- Selección – un poco mejor
  - Útil si: numero de datos es pequeño y si la operación de intercambio consume mucho tiempo comparado con las comparaciones
- Inserción – más versátil
  - El mejor en la mayoría de las situaciones
  - Aún para grandes cantidades de datos altamente desordenados,
- Requerimientos de memoria no son tan altos para estos algoritmos