Data Structures - Simple Sorting

Dr. TGI Fernando [1] [2]

[1] Email: tgi.fernando@gmail.com
[2] URL: http://tgifernando.wordpress.com/

Simple Sorting

As soon as you create a database, the next thing you want to arrange its records in a particular order (sorting).

- Arranging names in alphabetical order.
- Students by their index number.
- Home sales by price.
- Countries by GDP.

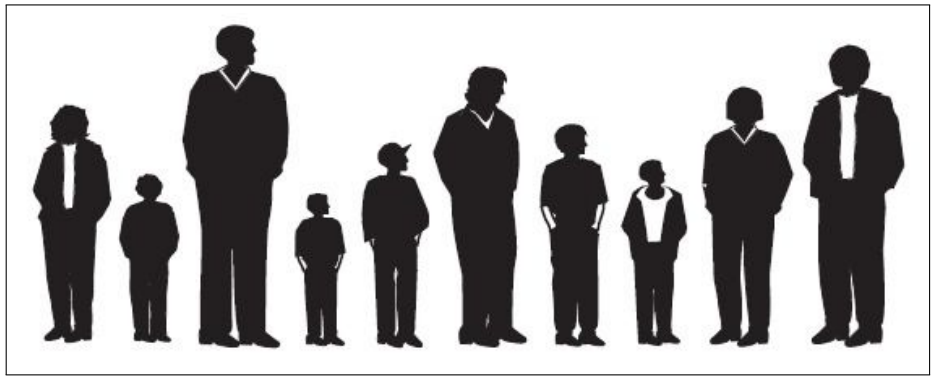
We will discuss three simple sorting algorithms.

1. Bubble sort
2. Selection sort
3. Insertion sort

} Simple, comparatively slow and better in some cases

How would you do it?

Suppose that we want to arrange the following baseball team in order of increasing height.

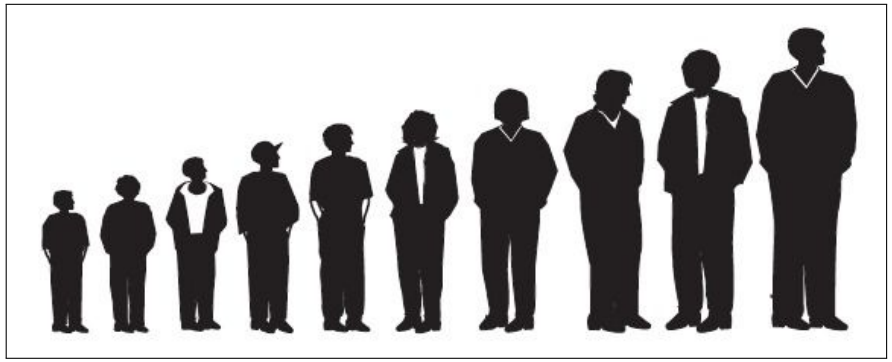


How would you do this sorting process?

Human

- A human can see all these players at once, he can pick out the tallest player instantly.
- No need to measure and compare everyone.
- Players can push a little to make room, and stand behind or in front of each other.
- After some ad hoc rearranging, a human can line up all the players.
- All involved in this process are intelligent.

The ordered baseball team



Computer Program

- Isn't able to glance over the data as humans.
- Can compare only two players (two values) at one time.
- Not intelligent - the program can't see the big picture.
- Must follow some simple rules.

Three algorithms - Bubble, Selection and Insertion

The three algorithms discuss here involve two steps, executed over and over until the data is sorted.

1. Compare two items.
2. Swap two items, or copy one item.

However, each algorithm handles the details in a different way.

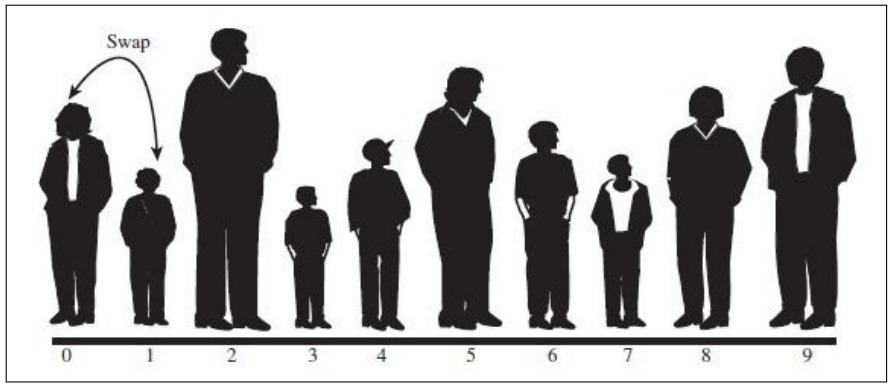
Bubble sort

Slow, simple and good beginning for the exploration of sorting methods.

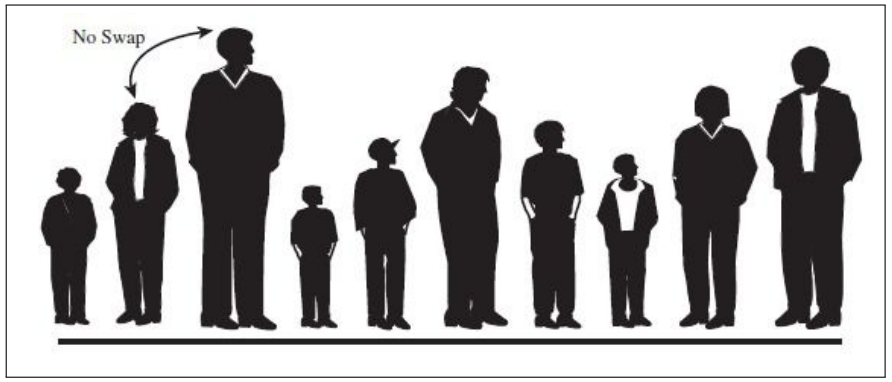
Let's assume there are N players, and the positions they're standing in are numbered from 0 on the left to $N-1$ on the right.

- Start at the left end of the line and compare the two players in positions 0 and 1.
  - If the one on the left (in 0) is taller, you swap them.
  - Else, you don't do anything.
- Move over one position, and compare the players in positions 1 and 2.
  - If the one on the left (this time, in 1) is taller, you swap them.
  - Else, you don't do anything.
- ...

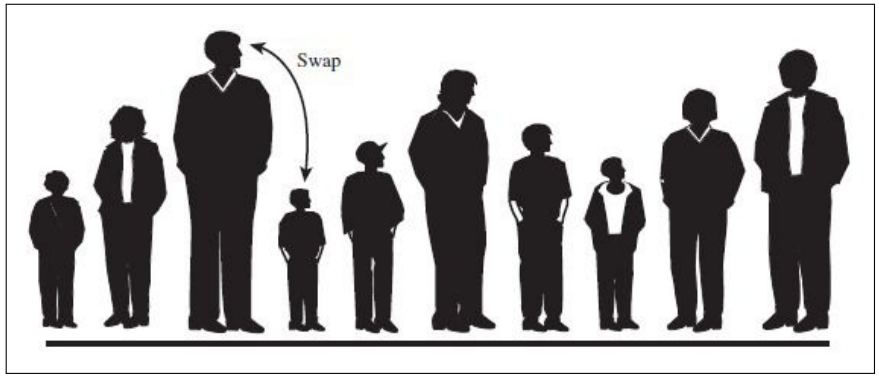
Bubble sort (Contd.)



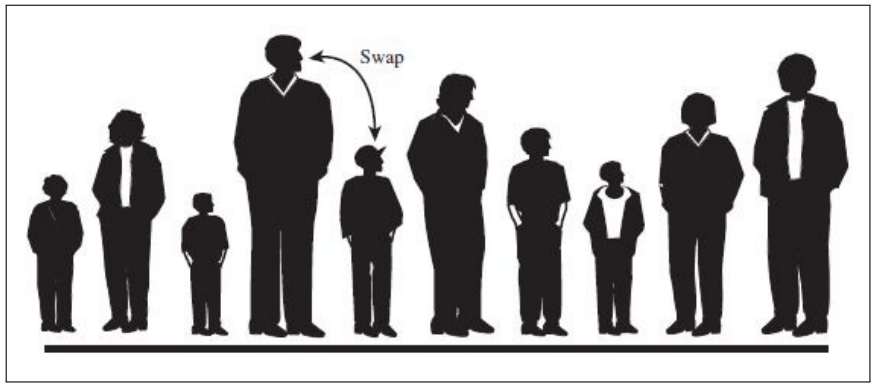
Bubble sort (Contd.)



Bubble sort (Contd.)



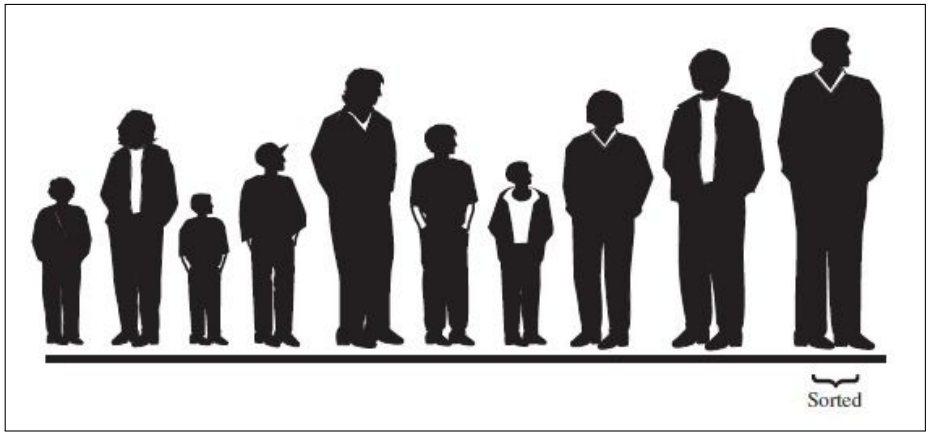
Bubble sort (Contd.)



Rules

1. Compare two players.
2. If the one on the left is taller, swap them.
3. Move one position right.
   You continue down the line this way until you reach the right end (First pass).
   As the algorithm progresses, the biggest items "bubble up" to the top end of the array.
   First pass: $N-1$ comparisons and somewhere between 0 and $N-1$ swaps
   The item at the end of the array is sorted and won't be moved again.
4. When you reach the first sorted player, start over at the left end of the line.
   However, this time you can stop one player short of the end of the line, at position $N-2$, because you know the last position, at $N-1$, already contains the tallest player.

After the first pass



Demonstration: For a demonstration of bubble sort, visit to Lafore's Bubble Sort Workshop Applet.

`BubbleArray` class

```
class BubbleArray {
  private long[] a; // ref to array a
  private int nElems; // number of data items
  //--------------------------------------------
  public BubbleArray (int max) { //constructor
    a = new long[max]; // create the array
    nElems = 0; // no items yet
  }
  //--------------------------------------------
  public void insert(long value) // put element into array
  {
    a[nElems] = value; // insert it
    nElems++; // increment size
  }
  //--------------------------------------------
  public void display () {...} // display array contents
  private void swap (int one, int two) {...} // swaps array
   elements in a[one] and a[two]
  public void bubbleSort () {...}
}
```

Efficiency of the Bubble Sort

Comparisons

If there are N number of elements in the array,
there are $N-1$ comparisons on the first pass
there are $N-2$ comparisons on the second pass
...

$\therefore$ the total number of comparisons (T)

$$\begin{aligned} T &= (N-1)+(N-2)+(N-3)+\cdots+1 \\ &= \frac{N(N-1)}{2} \\ &\approx \frac{N^2}{2} \text{ when } N \text{ is very large} \end{aligned}$$

Efficiency of the Bubble Sort (Contd.)

Swaps

If there are N number of elements in the array,
The average no of swaps on the first pass (S_1)

$$\begin{aligned} S_1 &= \frac{[0+1+2+\cdots+(N-1)]}{N} \\ &= \frac{N(N-1)}{2N} \\ &= \frac{N-1}{2} \end{aligned}$$

Similarly
The average no of swaps on the second pass (S_2) $= \frac{N-2}{2}$
The average no of swaps on the third pass (S_3) $= \frac{N-3}{2}$
$\cdots$

Efficiency of the Bubble Sort (Contd.)

$\therefore$ The average no of swaps (S)

$$\begin{aligned} S &= S_1 + S_2 + S_3 + \cdots \\ &= \frac{N-1}{2} + \frac{N-2}{2} + \frac{N-3}{2} + \cdots + \frac{1}{2} \\ &= \frac{N(N-1)}{4} \\ &\approx \frac{N^2}{4} \text{ when } N \text{ is very large} \end{aligned}$$

Both swaps and comparisons are proportional to N^2. Because constants don't count in Big O notation, we can ignore the 2 and the 4 and say that the bubble sort runs in O(N^2) time.