Bubble Sort

Con questo secondo metodo, si confrontano gli elementi del vettore a due a due. Se si trovano due elementi in ordine "scorretto", essi vengono scambiati di posto. Per esempio nel nostro caso, la prima esecuzione del ciclo produrrà gli scambi seguenti:

Example:
First Pass:
(**5 1** 4 2 8) –> (**1 5** 4 2 8), Here, algorithm compares the first two elements, and swaps since 5 > 1.
(1 **5 4** 2 8) –> (1 **4 5** 2 8), Swap since 5 > 4
(1 4 **5 2** 8) –> (1 4 **2 5** 8), Swap since 5 > 2
(1 4 2 **5 8**) –> (1 4 2 **5 8**), Now, since these elements are already in order (8 > 5), algorithm does not swap them.

Second Pass:
(**1 4** 2 5 8) –> (**1 4** 2 5 8)
(1 **4 2** 5 8) –> (1 **2 4** 5 8), Swap since 4 > 2
(1 2 **4 5** 8) –> (1 2 **4 5** 8)
(1 2 4 **5 8**) –> (1 2 4 **5 8**)
Now, the array is already sorted, but our algorithm does not know if it is completed. The algorithm needs one **whole** pass without **any** swap to know it is sorted.

Come si può notare alle volte si potrebbe interrompere prima del termine…..

Third Pass:
(**1 2** 4 5 8) –> (**1 2** 4 5 8)
(1 **2 4** 5 8) –> (1 **2 4** 5 8)
(1 2 **4 5** 8) –> (1 2 **4 5** 8)
(1 2 4 **5 8**) –> (1 2 4 **5 8**)

N=5
Posti 0,1,2,3,4

```
for (i = 0; i < n-1; i++)      // l'ultimo andrà a posto da se
    {for (j = 0; j < n-i-1; j++)
              {  if (v[j] > v[j+1])
                    { temp = v[j+1];
                      v[j+1] = v[j];
                      v[j] = temp;
                    }
              }
   }
```

Questo algoritmo non è molto efficiente e i suoi tempi di calcolo crescono al crescere delle dimensioni del vettore. Tuttavia vi sono diverse varianti di questo algoritmo che consentono di migliorarne le prestazioni. Per esempio, nel caso di vettori già parzialmente ordinati, è possibile aumentare la velocità dell'algoritmo fermandolo quando non ci sono più scambi (quando questa condizione si verifica, il vettore è ovviamente ordinato).