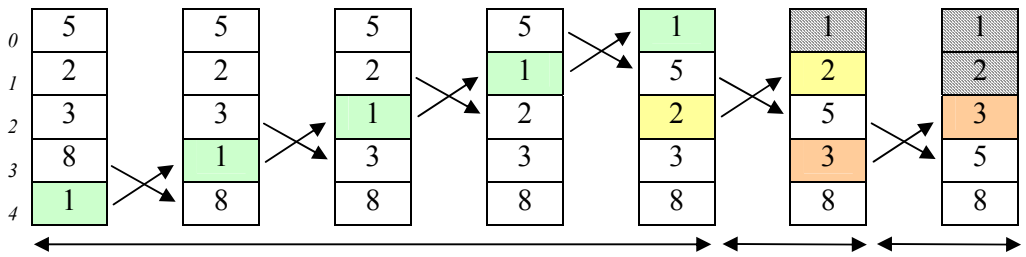
Corso di Ingegneria Elettronica
Insegnamento di "Strutture Software 1" a.a. 2005-06

Esercitazione n. 3

1. Sviluppare un metodo che implementi l'ordinamento a bolle (`bubbleSort`). L'array è scandito dall'indice di valore più alto a quello più basso, scambiando tra loro elementi adiacenti che sono fuori ordine. Per prima cosa si confronta `data[n-1]` con `data[n-2]`, scambiandoli se fuori ordine, poi si confronta `data[n-2]` con `data[n-3]`, scambiandoli se fuori ordine, e così via fino a `data[1]` e `data[0]`. In tal modo gli elementi minori sono fatti risalire fino agli indici di valore inferiore. Tuttavia questo è solo il primo passo attraverso l'array: si ripete nuovamente tale procedimento, ma l'ultimo confronto avviene tra `data[2]` e `data[1]`, perché l'elemento più piccolo è già nella sua posizione corretta, la posizione `0`. Questo secondo passo porta il secondo elemento più piccolo nella posizione `1`. La procedura continua fino all'ultimo passo, quando si effettua un solo confronto tra `data[n-1]` e `data[n-2]`, ed eventualmente uno scambio. Vediamo una rappresentazione grafica:



   Provare il corretto funzionamento dell'algoritmo in diverse situazioni: per esempio, array in ordine casuale, ordinato, ordinato in modo inverso e array di valori tutti uguali.

2. Calcolare la complessità computazionale asintotica dell'algoritmo di ordinamento a bolle.

3. Valutare sperimentalmente la complessità degli algoritmi `selectionSort` e `bubbleSort`. Introdurre nell'algoritmo delle variabili apposite, con le istruzioni di incremento nei punti opportuni, per contare il numero di confronti e spostamenti. Eseguire l'algoritmo su più istanze dei dati in ingresso, tracciando i grafici delle variabili contatore in funzione della dimensione dei dati in ingresso. Usare array di valori casuali di dimensione crescente: per esempio, $n=\{ 2^1, 2^2, 2^3, \ldots, 2^{14} \}$.