Documentazione di Matteo Provini e Fabio Sampietro 4I1

**BREVE INTRODUZIONE:**

L'**Insertion sort**, in italiano **ordinamento a inserimento**, è un algoritmo relativamente semplice per ordinare un array. Non è molto diverso dal modo in cui un essere umano, spesso, ordina un mazzo di carte. Esso è un algoritmo *in plac*e, cioè ordina l'array senza doverne creare una copia, risparmiando memoria. Pur essendo molto meno efficiente di algoritmi più avanzati, può avere alcuni vantaggi: ad esempio, è semplice da implementare ed è efficiente per insiemi di partenza che sono quasi ordinati.

**PSEUDOCODICE:**

Segue lo pseudocodice per l'algoritmo (si assume che la numerazione degli elementi negli array inizi da 0).

**function** insertionSort(array A)  
 **for** i ← 1 **to** length[A] **do**  
 value ← A[i]  
 j ← i-1  
 **while** j >= 0 **and** A[j] > value **do**  
 A[j + 1] ← A[j]  
 Te j ← j-1  
 A[j+1] ← value

**Tempo di calcolo nel caso peggiore:**

Il caso pessimo è invece quello in cui la sequenza di partenza sia ordinata al contrario. In questo caso, ogni iterazione dovrà scorrere e spostare ogni elemento della sottosequenza ordinata prima di poter inserire il primo elemento della sottosequenza non ordinata. Pertanto, in questo caso l'algoritmo di insertion sort ha complessità temporale quadratica, ossia O(n2)

**Tempo di calcolo nel caso migliore:**

Il caso ottimo per l'algoritmo è quello in cui la sequenza di partenza sia già ordinata. In questo caso, l'algoritmo ha tempo di esecuzione lineare. Infatti, in questo caso, in ogni iterazione il primo elemento della sottosequenza non ordinata viene confrontato solo con l'ultimo della sottosequenza ordinata.

**Lo spazio utilizzato nel caso peggiore:**

La struttura dati utilizzata per implementare l’algoritmo è un ARRAY monodimensionale, contenente i valori da ordinare, di conseguenza la complessità di spazio è O (n).