Soluzione Lab 8 – Es 4

Invece di procedere "per tentativi", cerchiamo di identificare un algoritmo che consenta di costruire una sequenza pessima per l'ordinamento mediante selezione.

Prendiamo come esempio una lista che contenga i valori 1, 2, 3, 4 e 5, in qualsiasi ordine iniziale.

Durante l'esecuzione di selection sort, prima dell'ULTIMA ricerca del minimo, le ultime due posizioni della lista contengono necessariamente i due valori maggiori della lista stessa, per effetto delle iterazioni precedenti dell'algoritmo: nelle ultime due posizioni ci sono, quindi, i numeri 4 e 5, in uno dei due ordini possibili: 4-5 oppure 5-4.
Perché avvenga uno scambio dopo quest'ultima ricerca di minimo, è necessario che la disposizione sia 5-4. Abbiamo quindi dimostrato che, per rendere pessime le prestazioni di selectionSort, i passi precedenti all'ultimo devono NECESSARIAMENTE disporre i valori nelle ultime due posizioni in modo che in essere vi sia la sequenza 5-4.

Analogamente, quando l'algoritmo arriva ad effettuare la PENULTIMA ricerca di minimo, nelle ultime tre posizioni della lista sono necessariamente presenti i tre valori maggiori della lista stessa, per effetto delle iterazioni precedenti dell'algoritmo. Ci dobbiamo ora chiedere quali siano le disposizioni di tali tre valori (3, 4 e 5) che danno luogo a uno scambio, al termine del quale viene prodotta la sequenza 3-5-4 (in modo da generare, nelle ultime due posizioni, la sequenza 5-4 che vi deve essere perché sia richiesto un ultimo scambio nell'iterazione successiva).
Dovendoci essere uno scambio, il valore 3 non può trovarsi inizialmente già nella prima delle ultime tre posizioni dell'array. Quali sono, quindi, le sequenze che, dopo uno scambio che coinvolga il valore 3, danno luogo all'unica sequenza finale utile, 3-5-4?
Ovviamente, il 3 può trovarsi in quella posizione finale corretta dopo essere stato scambiato con il 5, oppure con il 4. Quindi, le due uniche sequenze che, presentate inizialmente alla penultima iterazione dell'algoritmo, danno luogo al massimo numero di scambi sono:5-3-4 (scambiando poi il 3 con il 5) e 4-5-3 (scambiando poi il 3 con il 4).
In entrambi questi casi (e solo in questi casi) si effettua necessariamente uno scambio e, poi, viene generata la sequenza 3-5-4, che è l'unica a richiedere un ulteriore scambio all'ultima iterazione dell'algoritmo.

Procedendo a ritroso, ci dobbiamo ora chiedere quali disposizioni dei valori 2, 3, 4 e 5, presenti nelle ultime quattro posizioni della lista, diano luogo a uno scambio (per portare il valore 2 nella sua posizione finale corretta, la prima delle quattro posizioni in esame) e generino, dopo lo scambio, una delle due sequenze che ci interessano: 2-5-3-4 e 2-4-5-3. Per generare la prima di queste due mediante uno scambio che porti il valore 2 nella sua posizione corretta, si può partire solamente da una di queste tre situazioni: 5-2-3-4, 3-5-2-4, 4-5-3-2. Per generare la seconda, si può analogamente partire soltanto da una di queste altre tre situazioni: 4-2-5-3, 5-4-2-3, 3-4-5-2.

Infine, ci dobbiamo chiedere da quali situazioni iniziali si possa pervenire a una delle sei situazioni così generate, con il valore 1 nella prima posizione, effettuando necessariamente uno scambio. Con analoghe considerazioni, si ottengono 24 configurazioni, 4 per ciascuna delle 6 appena viste. Dalla prima (1-5-2-3-4), si ottengono, ad esempio, queste: 5-1-2-3-4, 2-5-1-3-4, 3-5-2-1-4, 4-5-2-3-1.

Abbiamo quindi identificato uno schema progettuale (cioè un algoritmo) per costruire tutte le sequenze iniziali che, ordinate con selectionSort, richiedono il massimo numero di scambi. Si può facilmente vedere che, nella disposizione di n valori distinti, sono possibili (n-1)! sequenze pessime per selectionSort.

Quella che è forse più facile da ricordare è la prima che abbiamo ottenuto, 5-1-2-3-4, che può essere generalizzata dicendo che presenta nella prima posizione il valore massimo della lista, seguito da tutti gli altri valori in ordine crescente.

Quindi, UNA situazione iniziale pessima per selectionSort che debba ordinare, ad esempio, l'insieme di valori 13, 6, 7, 10, 12, 24, è: 24 6 7 10 12 13

Una dimostrazione di questo tipo si chiama "costruttiva": mentre dimostra che un algoritmo risolve un determinato problema, costruisce anche l'algoritmo stesso!