Università Roma Tre – Dipartimento di Matematica e Fisica

Percorso Abilitante Speciale
Classe A048 – Matematica Applicata

Corso di Informatica

Algoritmi di ordinamento

Marco Liverani
(liverani@mat.uniroma3.it)

Sommario

- Il problema dell'ordinamento
- Approccio "cieco"
- Approccio elementare: ordinamento per selezione
- Approccio più ragionato: ordinamento a bolle
- Approccio ottimale: ordinamento per fusione

Il problema dell'ordinamento (sort) 1

- Dato un insieme di n elementi
  $$X = \{x_1, x_2, ..., x_n\}$$
  si chiede di trovare una permutazione
  $$\pi(X) = (x_{k1}, x_{k2}, ..., x_{kn})$$
  tale che $x_{k_i} \leq x_{k_{i+1}}$ per i=1, 2, ..., n–1
- Esempio: X = {34, 17, 5, 42, 21, 15, 34, 8}
  n = $|X|$ = 8
  Si vuole ottenere la permutazione di X
  $\pi(X)$ = (5, 8, 15, 17, 21, 34, 34, 42)
- È un po' come ordinare un mazzo di carte disordinato...

Il problema dell'ordinamento (sort) 2

- Non abbiamo alcuna informazione sull'insieme X da ordinare
- Procederemo quindi con una sequenza di confronti tra gli elementi di X e di scambi di posizione nella sequenza tra coppie di elementi, fino ad ottenere la *permutazione* ordinata degli elementi di X

Sommatorie...

- Ci sarà presto utile sapere quanto fa la somma dei primi n naturali

$$1+2+3+\ldots+(n-1)+n = ???$$

- Gauss l'ha scoperto a 12 anni!

$$\begin{array}{l} 1 + 2 + 3 + 4 + \ldots + n-1 + n + \\ n + n-1 + n-2 + n-3 + \ldots + 2 + 1 = \\ \hline n+1 + n+1 + n+1 + n+1 + \ldots + n+1 + n+1 = \\ = n\,(n+1) \end{array}$$

Quindi:

$$1+2+3+\ldots+(n-1)+n = n(n+1)/2$$

Esempio: $1+2+3+4+5 = 15 = 5\times 6/2$

Algoritmo completamente "cieco"

- Produciamo tutte le permutazioni degli elementi di X e verifichiamo se ogni permutazione è ordinata, fino a quando non ne avremo trovata una che risolve il problema
- Quante sono le permutazioni degli elementi di un insieme di cardinalità n?
- Ho n scelte per il primo elemento, per ogni scelta del primo elemento ho $n-1$ scelte per il secondo, poi $n-2$ scelte per il terzo...
- In totale: $n\times(n-1)\times(n-2)\times \ldots \times 2\times 1 = n!$
- Se n=15 dobbiamo provare $n!$=1.307.674.368.000 permutazioni. Calcolando 1.000.000 di permutaz. al secondo impiegheremmo circa 15 giorni... c'è un metodo *più efficiente*?

Ordinamento per selezione 1

- Scorro l'insieme X più volte e ad ogni scansione seleziono l'elemento più piccolo, lo tolgo dall'insieme e lo aggiungo alla sequenza Y ordinata
- Esempio:
  1. $X = \{8, 3, 5, 7, 1\}$, minimo = 1 ➔ $Y = (1)$
  2. $X = \{8, 3, 5, 7\}$, minimo = 3 ➔ $Y = (1, 3)$
  3. $X = \{8, 5, 7\}$, minimo = 5 ➔ $Y = (1, 3, 5)$
  4. $X = \{8, 7\}$, minimo = 7 ➔ $Y = (1, 3, 5, 7)$
  5. $X = \{8\}$, minimo = 8 ➔ $Y = (1, 3, 5, 7, 8)$
  6. $X = \emptyset$ ➔ $Y = (1, 3, 5, 7, 8)$ sequenza ordinata!

Ordinamento per selezione 2

1. leggi $x_1, x_2, \ldots, x_n$
2. sia i=1
3. se i>n vai al passo 11
4. sia j=i+1
5. se j>n vai al passo 9
6. se $x_j < x_i$ allora scambia x_i e x_j
7. sia j = j+1
8. vai al passo 5
9. sia i = i+1
10. vai al passo 3
11. scrivi $x_1, x_2, \ldots, x_n$
12. stop

Quante volte ripeto il confronto tra x_j e x_i?
$n-1 + n-2 + n-3 + \ldots + 2 + 1 = n(n-1)/2$

Quindi $\boldsymbol{O(n^2)}$

Ordinamento per selezione 3

- L'algoritmo "SELECTION SORT" è molto elementare e non adotta nessuna strategia per rendere più efficiente la ricerca della soluzione
- Ad esempio, se X è *già ordinato*, l'algoritmo non se ne accorge ed esegue ugualmente tutte le operazioni del procedimento...

Si può fare di meglio?

Ordinamento a bolle 1

- Confronto a due a due gli elementi adiacenti
- se l'ordine reciproco non è corretto li scambio
- ripeto il procedimento fino a quando non eseguo più nessuno scambio

Esempio: $X = \{8, 3, 5, 7, 1\}$

1. 8 3 5 7 1 → 3 8 5 7 1 → 3 5 8 7 1 → 3 5 7 8 1 → 3 5 7 1 8
2. 3 5 7 1 8 → 3 5 7 1 8 → 3 5 7 1 8 → 3 5 1 7 8
3. 3 5 1 7 8 → 3 5 1 7 8 → 3 1 5 7 8
4. 3 1 5 7 8 → 1 3 5 7 8

Soluzione!

Ordinamento a bolle 2

1. leggi $x_1, x_2, ..., x_n$
2. sia i=1, $flag$ = 0
3. se i>n o $flag$=1vai al passo 11
4. sia j=1, $flag$=1
5. se j>n–i vai al passo 9
6. se $x_i > x_{i+1}$ allora scambia x_i e x_{i+1} $flag$ = 0
7. sia $j = j$+1
8. vai al passo 5
9. sia $i = i$+1
10. vai al passo 3
11. scrivi $x_1, x_2, ..., x_n$
12. stop

Quante volte ripeto il confronto tra x_i e x_{i+1}? $n–1 + n–2 + n–3 + ... + 2 + 1 = n(n–1)/2$

Quindi $O(n^2)$

se $flag = 0$ allora ho eseguito almeno uno scambio: si deve ripetere la scansione della sequenza

Ordinamento a bolle 3

- L'algoritmo BUBBLE SORT ha la stessa complessità di SELECTION SORT: $O(n^2)$
- Nella pratica è più efficiente: sfrutta come condizione favorevole l'insieme già completamente o parzialmente ordinato
  - Caso peggiore: X è inizialmente ordinato al contrario ➔ $O(n^2)$
  - Caso migliore: X già ordinato ➔ $O(n)$

Ordinamento per fusione 1

- Suddivido X in n sequenze da un elemento ciascuno (sono ordinate!)
- Le fondo a due a due ottenendo la metà delle sequenze, lunghe il doppio, ma ordinate
- Ripeto l'operazione di fusione fino ad ottenere una sola sequenza di n elementi completamente ordinata

Ordinamento per fusione 1

- Esempio: $X = \{4, 7, 3, 1, 5, 2, 8, 6\}$