Algoritmi di Ordinamento

Luca Abeni

Esempio: Ordinamento di Numeri

- A cosa servono i tipi di dato strutturati? Non bastano i tipi scalari?
  - Capiamolo con un esempio...
- Problema: dato un insieme di numeri naturali, ordinarli
- Possibile soluzione: Ordinamento per Inserimento Diretto
  1. Scorrere tutti i numeri dal secondo all'ultimo
  2. Per ogni numero considerato, inserirlo nel giusto ordine fra i precedenti
- Descrizione molto informale...
  - "*Scorrere tutti i numeri*"?
  - "*Per ogni numero considerato*"?
  - "*Inserire nel giusto ordine*"?

Ordinamento per Inserimento - Esempio

- Cerchiamo di capire meglio l'ordinamento per inserimento...
- ...tramite un esempio!

44	55	12	42	94	18	6	67
44	55	12	42	94	18	6	67
44	55	12	42	94	18	6	67
12	44	55	42	94	18	6	67
12	42	44	55	94	18	6	67
12	42	44	55	94	18	6	67
12	18	42	44	55	94	6	67
6	12	18	42	44	55	94	67
6	12	18	42	44	55	67	94

Formalizziamo l'Algoritmo - 2

- Contatore `i`: posizione del numero che stiamo considerando
- Per ogni valore di `i` (da 2 a n), considera i numeri con posizione $<$ `i`
  - Altro contatore `j`, che va da `i - 1` a 1 (conta indietro)
- Se il numero in posizione `j` è $>$ del numero in posizione `i`, sposta tale numero a destra
- Se il numero in posizione `j` è $<$ del numero in posizione `i`, ferma il contatore `j` ed inserisci in posizione `j + 1` il numero che era in posizione `i`

Formalizziamo l'Algoritmo - 1

- Variabili: i, j. Insieme di definizione: numeri naturali
- Inoltre, appoggio (intero) e numeri da ordinare...
- `i = 2`
- mentre `i` $\leq n$
  - `appoggio` = numero in posizione `i`
  - `j = i - 1`
  - mentre `j` > 0 e numero in posizione `j` $>$ `appoggio`
    - numero in posizione `j + 1` = numero in posizione `j`
    - `j = j - 1`
  - `i = i + 1`
  - numero in posizione `j + 1` = `appoggio`

Formalizziamo l'Algoritmo - 3

- "numero in posizione `i` / `j`"?
  - Formalizziamo un po' meglio questa cosa...
- Usiamo un nome per identificare tutti i numeri...
  - Se un numero ha insieme di definizione $\mathcal{Z}$, questa variabile ha insieme di definizione $\mathcal{Z}^n$
- ...ed identifichiamo il singolo numero tramite un indice
  - In matematica, $\texttt{numero}_i$, $\texttt{numero}_j$
  - Ma usiamo qualche simbolo che sia meglio rappresentabile sui computer
  - `numero[i]`, `numero[j]`

Formalizziamo l'Algoritmo - 4

- Variabili: `i`, `j` $\in \mathcal{N}$; `appoggio` $\in \mathcal{Z}$ e `numero` $\in \mathcal{Z}^n$
- `i = 2`
- mentre `i` $\leq n$
  - `appoggio` $=$ `numero[i]`
  - `j = i - 1`
  - mentre `j` > 0 e `numero[j]` $>$ `appoggio`
    - `numero[j + 1] = numero[j]`
    - `j = j - 1`
  - `i = i + 1`
  - `numero[j + 1] = appoggio`

Array

- `numero` $\in \mathcal{Z}^n$
  - Variabile con insieme di definizione $\mathcal{Z}^n$
  - Composta da n variabili con insieme di definizione $\mathcal{Z}$ (variabili intere)
  - Singoli elementi accessibili come `numero[i]`
- Abbiamo appena scoperto gli Array!
- Array: tipo *strutturato* composto da variabili di un tipo semplice ripetute più volte
- Tutti gli elementi dell'array hanno lo stesso tipo

Ancora su Ordinamento di Numeri

- Problema: dato un insieme di numeri naturali, ordinarli
- Abbiamo visto Ordinamento per Inserimento Diretto
- Altra possibile soluzione: Ordinamento per Selezione Diretta
  1. Scorrere tutti i numeri dal primo al penultimo
  2. Per ogni numero considerato, selezionare l'elemento minimo fra i successivi e scambiarlo con l'elemento stesso
- Ancora, descrizione informale...
  - "*Scorrere tutti i numeri*"?
  - "*Per ogni numero considerato*"?
  - "*Selezionare l'elemento minimo*"?

Ordinamento per Selezione Diretta - Esempio

- Cerchiamo di capire meglio l'ordinamento per selezione diretta...
- ...tramite un esempio!

44	55	12	42	94	18	6	67
44	55	12	42	94	18	6	67
6	55	12	42	94	18	44	67
6	12	55	42	94	18	44	67
6	12	18	42	94	55	44	67
6	12	18	42	94	55	44	67
6	12	18	42	44	55	94	67
6	12	18	42	44	55	94	67
6	12	18	42	44	55	67	94

Formalizziamo l'Algoritmo - 1

- Contatore `i`: posizione del numero che stiamo considerando
- Per ogni valore di `i` (da 0 a $n-2$), considera i numeri con posizione $\geq$ `i`
  - Altro contatore `j`, che va da `i` (meglio: `i + 1`) a $n-1$
- Se il numero in posizione `j` è $<$ del minimo, tale numero diventa il minimo; ricorda la posizione del minimo
- Una volta trovato il minimo, va scambiato col numero in posizione `i`

Formalizziamo l'Algoritmo - 2

- Variabili: `i`, `j`, `imin` $\in \mathcal{N}$; `min` $\in \mathcal{Z}$ e `numero` $\in \mathcal{Z}^n$
- `i = 0`
- mentre `i` $< n - 1$
  - `min` $=$ `numero[i]`
  - `imin = i`
  - `j = i + 1`
  - mentre `j` $< n$
    - Se `numero[j]` $<$ `min`
      - min $=$ `numero[j]`
      - `imin = j`
    - `j = j + 1`
  - `numero[imin]` $=$ `numero[i]`
  - `numero[i]` $=$ `min`
  - `i = i + 1`

Confronto con Inserimento Diretto

- Inserimento diretto trova la posizione corretta per `numero[i]` fra i numeri a sinistra
- Selezione diretta trova l'elemento dell'array corretto per una determinata posizione `i`, considerando i numeri a destra
  - Inserimento diretto: `i` va dal secondo all'ultmo elemento dell'array (`i` rappresenta l'elemento da spostare)
  - Selezione diretta: `i` va dal primo al penultmo elemento dell'array (`i` rappresenta la posizione in cui spostare un elemento)

Altro Algoritmo di Ordinamento di Numeri

- Problema: dato un insieme di numeri naturali, ordinarli
- Abbiamo visto Ordinamento per Inserimento Diretto e Selezione Diretta
- Altra possibile soluzione: Ordinamento per Scambio Diretto
  1. Scorrere tutti i numeri dal secondo all'ultimo
  2. Se un numero ed il precedente sono nell'ordine sbagliato, scambiarli
  3. Ripetere questi due passi fino a che non ci sono più scambi
- Ancora, descrizione informale...
  - "*Scorrere tutti i numeri*"?
  - "*Ripetere fino a che non ci sono scambi*"?

Ordinamento per Scambio Diretto - Esempio

44	55	12	42	94	18	6	67
44	55	12	42	94	18	6	67
44	55	12	42	94	18	6	67
44	12	55	42	94	18	6	67
44	12	42	55	94	18	6	67
44	12	42	55	94	18	6	67
44	12	42	55	18	94	6	67
44	12	42	55	18	6	94	67
44	12	42	55	18	6	67	94
12	44	42	55	18	6	67	94
12	42	44	55	18	6	67	94
12	42	44	55	18	6	67	94
12	42	44	18	55	6	67	94
12	42	44	18	6	55	67	94
12	42	44	18	6	55	67	94

Ordinamento per Scambio Diretto - Esempio

12	[42]	44	18	6	55	67	94
12	42	[44]	18	6	55	67	94
12	42	44	[18]	6	55	67	94
12	42	18	44	[6]	55	67	94
12	42	18	6	44	[55]	67	94
12	42	18	6	44	55	[67]	94
12	42	18	6	44	55	67	[94]
[12]	[42]	18	6	44	55	67	94
12	42	[18]	6	44	55	67	94
12	18	42	[6]	44	55	67	94
12	18	6	42	[44]	55	67	94
12	18	6	42	44	[55]	67	94
12	18	6	42	44	55	[67]	94
12	18	6	42	44	55	67	[94]

Ordinamento per Scambio Diretto - Esempio

12	18	6	42	44	55	67	94
12	18	6	42	44	55	67	94
12	6	18	42	44	55	67	94
12	6	18	42	44	55	67	94
...							
12	6	18	42	44	55	67	94
6	12	18	42	44	55	67	94

Formalizziamo l'Algoritmo

- Variabili: `i` $\in \mathcal{N}$; `appoggio` $\in \mathcal{Z}$; `scambio` $\in \{Vero, Falso\}$ e `numero` $\in \mathcal{Z}^n$
- ripeti
  - `i = 1`
  - `scambio = falso`
  - mentre `i` $< n$
    - Se `numero[i - 1]` $>$ `numero[i]`
      - `appoggio` $=$ `numero[i]`
      - `numero[i]` $=$ `numero[i - 1]`
      - `numero[i - 1]` $=$ `appoggio`
      - `scambio = vero`
    - `i = i + 1`
- fino a che `scambio` $=$ falso

Considerazioni sull'Algoritmo

- L'algoritmo di Scambio Diretto richiede varie iterazioni
  - Ogni iterazione scorre tutti i numeri dal secondo all'ultimo
- Ogni iterazione tende a "spingere" i numeri più grandi verso destra
- Dopo i iterazioni, gli i numeri più grandi saranno alla fine dell'array
- L'algoritmo è generalmente noto come Bubble Sort
  - Analogia fra bolle che salgono verso l'alto e numeri che vengono spinti verso destra...