Fondamenti di Informatica (Elettronici)

Algoritmi fondamentali: merge – Lezione 32

21 dicembre 2020

Algoritmi fondamentali: fusione

Section 1

Algoritmo di fusione (merge)

Algoritmo di fusione

Siano date due tavole memorizzate come array, e siano ordinate per valori crescenti della chiave (valore).

La fusione delle due tavole consiste nel generare una unica tavola ordinata a partire dalle due tavole ordinate.

Algoritmo di fusione

Siano date due tavole memorizzate come array, e siano ordinate per valori crescenti della chiave (valore).

La fusione delle due tavole consiste nel generare una unica tavola ordinata a partire dalle due tavole ordinate.

Metodo

Scandire congiuntamente le due tabelle, utilizzando due indici;
ad ogni ciclo:

Algoritmo di fusione

Siano date due tavole memorizzate come array, e siano ordinate per valori crescenti della chiave (valore).

La fusione delle due tavole consiste nel generare una unica tavola ordinata a partire dalle due tavole ordinate.

Metodo

Scandire congiuntamente le due tabelle, utilizzando due indici;
ad ogni ciclo:

- copiare nella tabella di uscita l'elemento di chiave minima

Algoritmo di fusione

Siano date due tavole memorizzate come array, e siano ordinate per valori crescenti della chiave (valore).

La fusione delle due tavole consiste nel generare una unica tavola ordinata a partire dalle due tavole ordinate.

Metodo

Scandire congiuntamente le due tabelle, utilizzando due indici;
ad ogni ciclo:

- copiare nella tabella di uscita l'elemento di chiave minima
- spostare uno degli indici

Algoritmo di fusione

Siano date due tavole memorizzate come array, e siano ordinate per valori crescenti della chiave (valore).

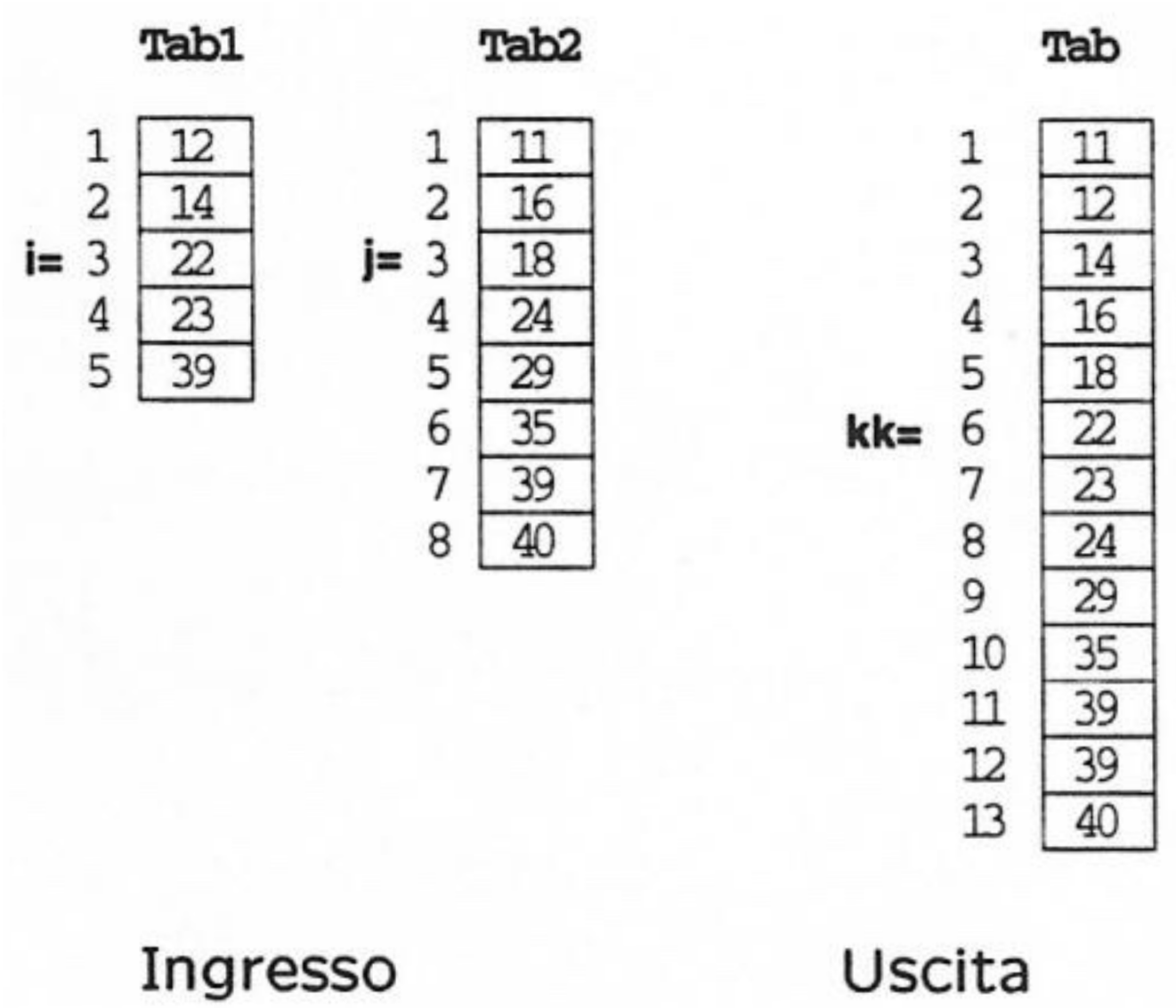
La fusione delle due tavole consiste nel generare una unica tavola ordinata a partire dalle due tavole ordinate.

Metodo

Scandire congiuntamente le due tabelle, utilizzando due indici;
ad ogni ciclo:

- copiare nella tabella di uscita l'elemento di chiave minima
- spostare uno degli indici
- quando la scansione di una tabella e' completata:
  esci dal ciclo e ricopia il resto dell'altra

Algoritmo di fusione



Section 2

Merge: Prima versione

Merge: Prima versione

Pseudo-codice

```
function merge(tab1::Array, tab2::Array)
    # 1. inizializzazioni:
            n1 = length(tab1)     # lunghezza di tab1
            n2 = length(tab2)     # lunghezza di tab2
            tab = Array{Int,1}() # array di uscita (vuota)
            i = j = 1;            # indici dei due array
    # 2. cicla finché una delle tabelle non termina:
            # copiando in uscita l'elemento minore
            # spostando l'indice dell'elemento copiato
    # 3. copia in output la parte residua
            # della tabella non terminata
end
```

Section 3

Merge: Versione finale

Algoritmo di fusione

```
julia> function merge(tab1::Array, tab2::Array, debug=false)
          n1 = length(tab1); n2 = length(tab2)
          tab = Array{Int,1}()
          i = j = 1;
          while i <= n1 && j <= n2
             if tab1[i] <= tab2[j]
                push!(tab, tab1[i]);  i += 1
             else
                push!(tab, tab2[j]);  j += 1
             end;              if debug @show tab end
          end
          if i-1 == n1
             tab = append!(tab, tab2[j:n2])
          else
             tab = append!(tab, tab1[i:n1])
          end;              if debug @show tab end
          return tab
        end
merge (generic function with 1 method)
```

Algoritmo di fusione – tracing

```
julia> merge( sort!(rand(-20:20, 5)), sort!(rand(-20:20, 13)), true )'
tab = [-18]
tab = [-18, -13]
tab = [-18, -13, -12]
tab = [-18, -13, -12, -12]
tab = [-18, -13, -12, -12, -11]
tab = [-18, -13, -12, -12, -11, -10]
tab = [-18, -13, -12, -12, -11, -10, -9]
tab = [-18, -13, -12, -12, -11, -10, -9, -7]
tab = [-18, -13, -12, -12, -11, -10, -9, -7, -6]
tab = [-18, -13, -12, -12, -11, -10, -9, -7, -6, -4, -3, -1, 10, 12, 13, 14, 15, 17]
1×18 LinearAlgebra.Adjoint{Int64,Array{Int64,1}}:
 -18  -13  -12  -12  -11  -10  -9  -7  -6  -4  -3  -1  10  12  13  14  15  17
```

Algoritmo di fusione – tracing

```
julia> merge( sort!(rand(-20:20, 5)), sort!(rand(-20:20, 13)), true )'
tab = [-18]
tab = [-18, -13]
tab = [-18, -13, -12]
tab = [-18, -13, -12, -12]
tab = [-18, -13, -12, -12, -11]
tab = [-18, -13, -12, -12, -11, -10]
tab = [-18, -13, -12, -12, -11, -10, -9]
tab = [-18, -13, -12, -12, -11, -10, -9, -7]
tab = [-18, -13, -12, -12, -11, -10, -9, -7, -6]
tab = [-18, -13, -12, -12, -11, -10, -9, -7, -6, -4, -3, -1, 10, 12, 13, 14, 15, 17]
1×18 LinearAlgebra.Adjoint{Int64,Array{Int64,1}}:
 -18  -13  -12  -12  -11  -10  -9  -7  -6  -4  -3  -1  10  12  13  14  15  17

julia> merge( sort!(rand(-20:20, 5)), sort!(rand(-20:20, 13)) )'
1×18 LinearAlgebra.Adjoint{Int64,Array{Int64,1}}:
 -18  -13  -12  -12  -11  -10  -9  -7  -6  -4  -3  -1  10  12  13  14  15  17
```

Algoritmo di fusione – esecuzione

1. Generazione di dati casuali di test

```
julia> tab = rand(-20:20, 5);

julia> tab'
1×5 LinearAlgebra.Adjoint{Int64,Array{Int64,1}}:
 -15  5  -16  -17  0
```

Algoritmo di fusione – esecuzione

1. Generazione di dati casuali di test

```
julia> tab = rand(-20:20, 5);

julia> tab'
1×5 LinearAlgebra.Adjoint{Int64,Array{Int64,1}}:
 -15  5  -16  -17  0
```

Algoritmo di fusione – esecuzione

1. Generazione di dati casuali di test

```
julia> tab = rand(-20:20, 5);

julia> tab'
1×5 LinearAlgebra.Adjoint{Int64,Array{Int64,1}}:
 -15  5  -16  -17  0
```

1. Ordinamento dei dati

```
julia> tab1 = sort(rand(-20:20, 5));

julia> tab1'
1×5 LinearAlgebra.Adjoint{Int64,Array{Int64,1}}:
 -18  -2  0  4  8

julia> tab2 = sort(rand(-20:20, 5));

julia> tab2'
1×5 LinearAlgebra.Adjoint{Int64,Array{Int64,1}}:
 -15  -7  -4  6  9
```

Algoritmo di fusione – esecuzione

1. Esecuzione fusione di due tabelle (array) ordinate(i)

```
julia> tab = merge(tab1,tab2);

julia> tab'
1×10 LinearAlgebra.Adjoint{Int64,Array{Int64,1}}:
 -18  -15  -7  -4  -2  0  4  6  8  9
```

Algoritmo di fusione – esecuzione

1. Esecuzione fusione di due tabelle (array) ordinate(i)

```
julia> tab = merge(tab1,tab2);

julia> tab'
1×10 LinearAlgebra.Adjoint{Int64,Array{Int64,1}}:
 -18  -15  -7  -4  -2  0  4  6  8  9
```

Algoritmo di fusione – esecuzione

1. Esecuzione fusione di due tabelle (array) ordinate(i)

```
julia> tab = merge(tab1,tab2);

julia> tab'
1×10 LinearAlgebra.Adjoint{Int64,Array{Int64,1}}:
 -18  -15  -7  -4  -2  0  4  6  8  9
```

Algoritmo di fusione – esempi

```
julia> tab1 = sort!(rand(-20:20, 5))'
1×5 LinearAlgebra.Adjoint{Int64,Array{Int64,1}}:
 -19  -14  -14  -13  6

julia> tab2 = sort!(rand(-20:20, 13))'
1×13 LinearAlgebra.Adjoint{Int64,Array{Int64,1}}:
 -20  -20  -12  -12  -11  -6  -3  0  2  5  10  11  19

julia> merge(tab1,tab2)'
1×18 LinearAlgebra.Adjoint{Int64,Array{Int64,1}}:
 -20  -20  -19  -14  -14  -13  -12  -12  -11  -6  -3  0  2  5  6  10  11  1
```

Algoritmo di fusione – esempi

```
julia> merge( sort!(rand(-20:20, 5)), sort!(rand(-20:20, 13)) )'
1×18 LinearAlgebra.Adjoint{Int64,Array{Int64,1}}:
 -17  -15  -12  -9  -4  -3  -2  1  1  3  4  7  7  10  14  17  17  20

julia> merge( sort!(rand(-20:20, 5)), sort!(rand(-20:20, 13)) )'
1×18 LinearAlgebra.Adjoint{Int64,Array{Int64,1}}:
 -20  -17  -17  -16  -15  -11  -11  -8  -5  -5  0  1  9  9  13  13  15  18

julia> merge( sort!(rand(-20:20, 5)), sort!(rand(-20:20, 13)) )'
1×18 LinearAlgebra.Adjoint{Int64,Array{Int64,1}}:
 -20  -17  -16  -14  -12  -10  -8  -6  -5  -4  -1  0  0  7  10  13  14  19
```

Section 4

Merge: complessità

Analisi di complessità 1/2

Complessita' di tempo

E' facile convincersi che la complessità dell'operazione di fusione è $O(n+m)$, dove n e m sono la misura dei due sottoinsiemi di dati d'ingresso.

La complessità di tempo, in questo caso, è pari alla somma delle due fasi (ciclo e copia della parte rimanente dell'input)

le operazioni dominanti sono le scritture (in tempo costante) dei singoli termini dentro la collezione d'uscita, di lunghezza $n = \texttt{n1}$ e $m = \texttt{n2}$, da cui

$$T(n,m) = \Theta(n+m)$$

.

Analisi di complessità 2/2

Complessita' di spazio

E' pari alla misura dell'output, se non si riscrive su una delle collezioni di ingresso:

$$space(n, m) = \Theta(n + m)$$

.