Algoritmi e Strutture Dati

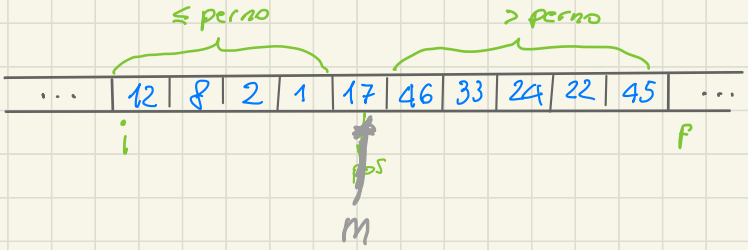
Lezione 12

24 ottobre 2022


≤ perno
> perno
. . .
perno
. . .

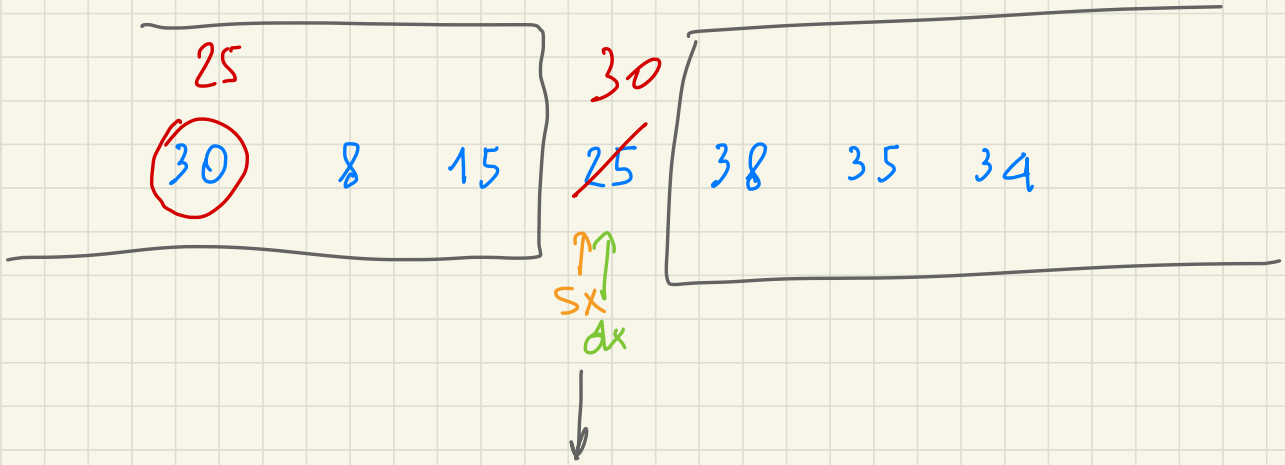
```
ALGORITMO quickSort (Array A[0..n-1])
    quickSort (A, 0, n)
```


```
PROCEDURA quickSort (Array A, indice i, indice f)
    IF f - i > 1 THEN
        m ← particiona (A, i, f)
        quickSort (A, i, m)
        quickSort (A, m+1, f)
```



1 2 8 12 17 22 24 33 45 46

```
ALGORITMO partiziona (Array A, indice i, indice f) → indice
    perno ← A[i]
    dx ← f, sx ← i
    WHILE sx < dx DO
        DO dx ← dx - 1 WHILE A[dx] > perno
        DO sx ← sx + 1 WHILE sx < dx AND A[sx] ≤ perno
        IF sx < dx THEN
            Scambia A[sx] con A[dx]
    scambia A[i] con A[dx]
    RETURN dx
```

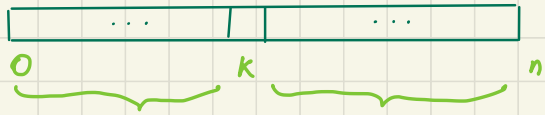


QuickSort: numero di confronti

```
PROCEDURA quickSort (Array A, indice i, indice f)
  IF f-i > 1 THEN
    m ← partiziona (A, i, f)
    quickSort (A, i, m)
    quickSort (A, m+1, f)
```

$$C(n) = \begin{cases} 0 & \text{se } n \leq 1 \\ n + C(k) + C(n-k-1) & \text{altrimenti} \end{cases}$$

k ?



$$C_w(n) = \begin{cases} 0 & \text{se } n \leq 1 \\ n + \max \{ C(k) + C(n-k-1) \mid k = 0..n-1 \} & \text{altrimenti} \end{cases}$$

$\hookrightarrow C_w(n) = \Theta(n^2)$ caso peggiore

$$C_b(n) = \begin{cases} 0 & \text{se } n \leq 1 \\ n + \min \{ C(k) + C(n-k-1) \mid k = 0..n-1 \} & \text{altrimenti} \end{cases}$$

$C_b(n) \simeq n \log_2 n$

#cfr nel caso medio

$$C(n) = \begin{cases} 0 & \text{se } n \le 1 \\ n + C(k) + C(n-k-1) & \text{altrimenti} \end{cases}$$

$$C(n) = \begin{cases} 0 & \text{se } n \le 1 \\ \dfrac{\sum_{k=0}^{n-1} n + C(k) + C(n-k-1)}{n} & \text{altrimenti} \end{cases}$$

$$C(n) = \frac{1}{n}\sum_{k=0}^{n-1} n + \frac{1}{n}\sum_{k=0}^{n-1} C(k) + \frac{1}{n}\sum_{k=0}^{n-1} C(n-k-1)$$

$\frac{1}{n}\sum_{k=0}^{n-1} n = n$

$\sum_{k=0}^{n-1} C(k) = C(0) + C(1) + \dots C(n-1)$

$\sum_{k=0}^{n-1} C(n-k-1) = C(n-1) + C(n-2) + \dots + C(1) + C(0)$

$=$

$$= n + \frac{2}{n}\sum_{i=0}^{n-1} C(i) = n + \frac{2}{n}\sum_{i=2}^{n-1} C(i)$$

$C(0) = C(1) = 0$

$$C(n) = \begin{cases} 0 & \text{se } n \le 1 \\ n + \frac{2}{n} \sum_{i=2}^{n-1} C(i) & \text{altrimenti} \end{cases}$$

Dimostriamo che

$$C(n) \le 2n \ln n \quad \text{per } n \ge 1$$

Per induzione

Base $n = 1$ $\quad C(1) = 0 = 2 \cdot 1 \ln 1$

Induzione

$$C(n) = n + \frac{2}{n} \sum_{i=2}^{n-1} C(i) \le n + \frac{2}{n} \sum_{i=2}^{n-1} 2i \ln i = n + \frac{4}{n} \sum_{i=2}^{n-1} i \ln i$$

ip ind. $C(i) \le 2i \ln i$

$$\le n + \frac{4}{n} \left[\frac{n^2}{2} \ln n - \frac{n^2}{4} \right]$$

$$\sum_{i=1}^{n-1} i \ln i \le \frac{n^2}{2} \ln n - \frac{n^2}{4}$$

$$= n + 2n \ln n - n = 2n \ln n$$

$$C(n) \le 2n \ln n \approx 1.39 n \log_2 n$$ # cfr ntle caso med.

CONFRONTI

Caso peggiore $\approx \frac{n^2}{2}$

Caso migliore $\approx n \log_2 n$

Caso medio $\approx 1.39\, n \log_2 n$

QuickSort: uso dello spazio

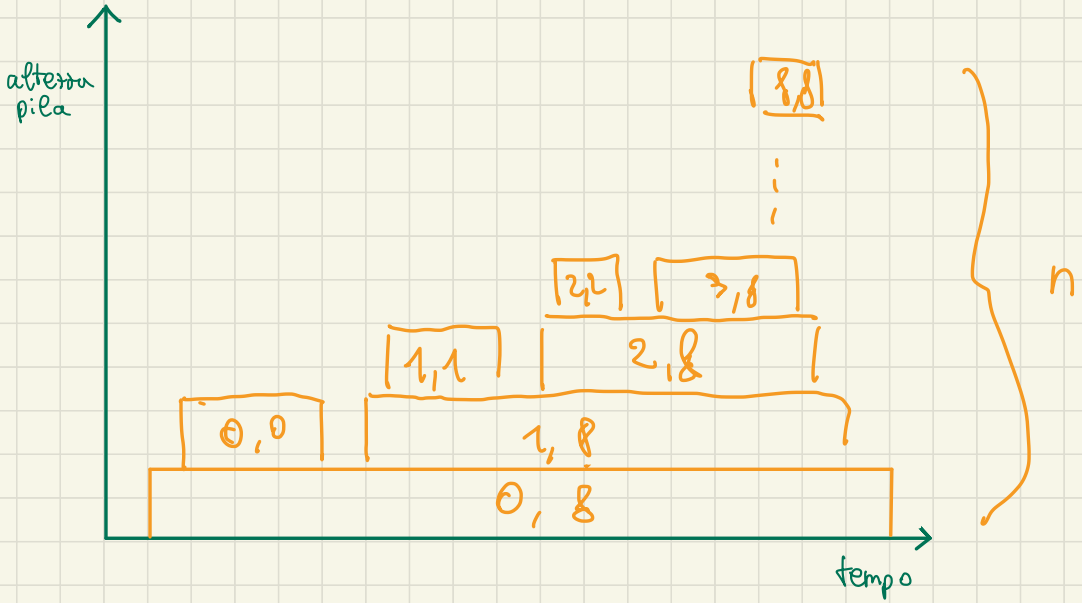
```
ALGORITMO quickSort (Array A[0..n-1])
    quickSort (A, 0, n)

PROCEDURA quickSort (Array A, indice i, indice f)
    IF f - i > 1 THEN
        m ← partiziona (A, i, f)
        quickSort (A, i, m)
        quickSort (A, m+1, f)
```

```
ALGORITMO partiziona (Array A, indice i, indice f) → indice
    perno ← A[i]
    sx ← i, dx ← f
    WHILE sx < dx DO
        DO dx ← dx - 1 WHILE A[dx] > perno
        DO sx ← sx + 1 WHILE sx < dx AND A[sx] ≤ perno
        IF sx < dx THEN
            scambia A[sx] con A[dx]
    scambia A[i] con A[dx]
    RETURN dx
```

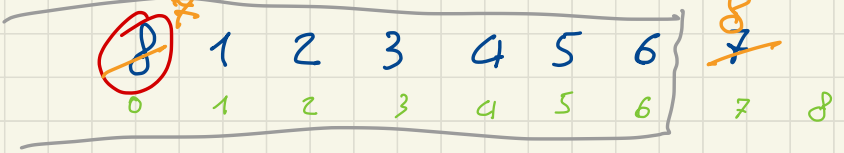
```
PROCEDURA quickSort (Array A, indice i, indice f)
    IF  f - i > 1  THEN
        m ← partiziona (A, i, f)
        quickSort (A, i, m)
        quickSort (A, m+1, f)
```

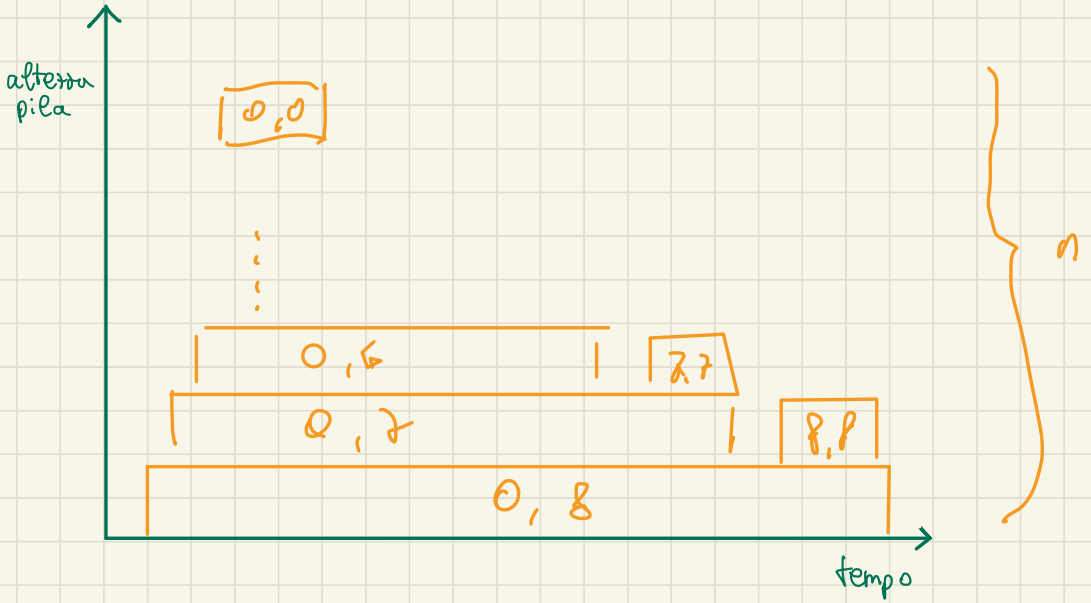
[A] Array già ordinato

1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8

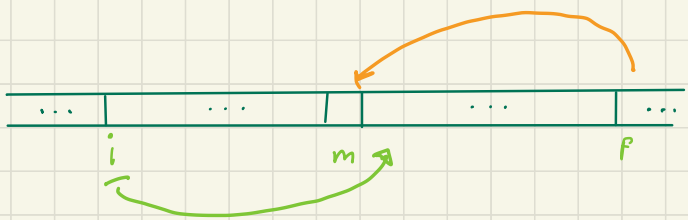


B Array quasi ordinato





```
PROCEDURE quickSort (Array A, indice i, indice f)
  IF f-i > 1 THEN
    m ← partition (A, i, f)
    quickSort (A, i, m)
    quickSort (A, m+1, f)
```




```
PROCEDURE quickSort (Array A, indice i, indice f)
  WHILE f-i > 1 DO
    m ← partition (A, i, f)
    quickSort (A, i, m)
    i ← m+1
```

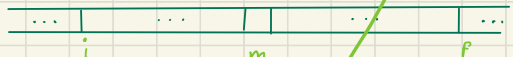
```
PROCEDURE quickSort (Array A, indice i, indice f)
  IF f-i > 1 THEN
    m ← partition (A, i, f)
    quickSort (A, m+1, f)
    quickSort (A, i, m)
```

```
PROCEDURE quickSort (Array A, indice i, indice f)
  WHILE f-i > 1 DO
    m ← partition (A, i, f)
    quickSort (A, m+1, f)
    f ← m
```

```
PROCEDURE quickSort (Array A, index i, index f)
    WHILE f-i > 1 DO
        m ← partition (A, i, f)
        quickSort (A, i, m)
        i ← m+1
```

```
PROCEDURE quickSort (Array A, index i, index f)
    WHILE f-i > 1 DO
        m ← partition (A, i, f)
        quickSort (A, m+1, f)
        f ← m
```

```
PROCEDURE quickSort (Array A, index i, index f)
    WHILE f-i > 1 DO
        m ← partition (A, i, f)
        IF m-i < f-m THEN
            quickSort (A, i, m)
            i ← m+1
        ELSE
            quickSort (A, m+1, f)
            f ← m
```



QuickSort : spazio

- versione migliorata

Altezza pila $\Theta(\lg n)$
Record di attivazione $\Theta(1)$
} → Spazio $\Theta(\lg n)$

- versione base

Altezza pila $\Theta(n)$
Record di attivazione $\Theta(1)$
} → Spazio $\Theta(n)$

Tipi di dati

COSA

Tipo di una variabile

Attributo che specifica l'insieme di *valori* che la variabile può assumere e le relative *operazioni*

classi Java

DATI ↔ oggetti

METODI

Esempio

Tipo Dizionario

Collezione di elementi ciascuno dei quali è caratterizzato da una *chiave*

Chiavi

Appartengono a un dominio *totalmente* ordinato

Operazioni tipiche

- inserimento
- ricerca
- cancellazione

Strutture dati

COME

Struttura dati

Specifica *organizzazione delle informazioni* che permette di realizzare e implementare un determinato tipo di dati

stesso tipo ≠ struttura

es dizionario

- array ordinato in base alla chiave
  - RICERCA $\Theta(\log n)$
  - INSERIMENTO $\Theta(n)$
- array non ordinato
  - RICERCA $\Theta(n)$
  - INSERIMENTO $O(1)$

Collezioni: strutture indicizzate

- Allocate in una *porzione contigua* di memoria
- Accesso mediante *indice* (posizione)
- Tempo di accesso *indipendente* dalla posizione del dato

Collezioni: strutture collegate

- Non è necessario allocare l'intera struttura in una porzione contigua di memoria
- Elementi collegati tra loro
- Passaggio da un elemento ad altri tramite collegamenti
- Varie tipologie di collegamento (liste, alberi, ...)

STRUTTURE DINAMICHE