Titolo nota 17/03/2016

QUICK_SORT

$\Theta(n^2)$ caso pessimo

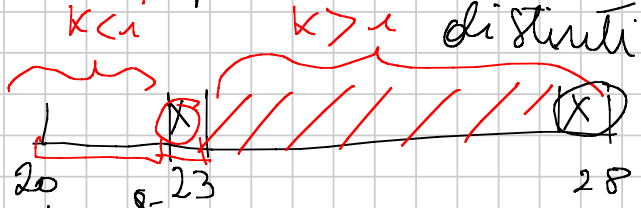
$O(n \log n)$ caso medio

Partition

Selezione del K-esimo el. Ipotesi: elementi distinti

soluzione banale:

1. ordino l'array
2. accedi all'elemento di posizione K?



$k = 7$

$i = q - p + 1$

$i = 23 - 20 + 1$

Titolo nota 17/03/2016

```
QUICK_Select (A, p, r, k)
if (p == r) return A[p]
else {
q = Partition (A, p, q, k);
i = q - p + 1;
if (k == i) return A[q];
    else if k < i return QUICK_Select (A, p, q-1, k);
         else return QUICK_Select (A, q+1, r, k-i);
}
```

caso pessimo:

Partition genera partizioni sbilanciate

$$T(n) = T(n-1) + \Theta(n)$$

$$T(n) = \Theta(n^2)$$

Titolo nota 17/03/2016

caso ottimo: partizioni bilanciate — MS spesso $\Theta(n)$ — QS spesso $\Theta(1)$

caso medio: $\Theta(n)$

$$T(n) = T\left(\frac{n}{2}\right) + \Theta(n)$$

$a = 1 \quad b = 2 \quad n^{\log_2 1} = n^0 = \Theta(1) = O(1) = K$

$n^1 : n^0$? $\quad n = \Omega(n^{0+\varepsilon})$ per $\varepsilon \leq 1$

caso 3

$$\boxed{T(n) = \Theta(n)}$$

se vale la condizione di regolarità.

$$a f\left(\frac{n}{b}\right) \leq c f(n) \qquad \frac{n}{2} \leq cn \quad \text{per} \quad c \geq \frac{1}{2}$$

Titolo nota 17/03/2016

Algoritmo A $\quad T(n) = 7\,T\left(\frac{n}{2}\right) + n^2$

Algoritmo A' $\quad T(n) = a\,T\left(\frac{n}{4}\right) + n^2$

Determinare il massimo valore di a per cui A' è più veloce di A.

A: $\quad T(n) = 7\,T\left(\frac{n}{2}\right) + n^2 \qquad a = 7 \quad b = 2 \quad n^{\log_2 7} \simeq n^{2.8}$

$n^2 : n^{2.8} \qquad n^2 = O\left(n^{2.8-\varepsilon}\right)$ per $\varepsilon \leq 0.8$

caso 1 : $T(n) = \Theta\left(n^{2.8}\right)$

A': $\quad a \qquad b = 4 \qquad n^{\log_4 a} \qquad f(n) = n^2$

per $a = 16$ $\quad n^{\log_4 16} = n^2 : n^2 \qquad T(n) = \Theta\left(n^2 \log n\right)$ caso 2

per $a < 16$ $\quad n^2 = \Omega\left(n^{\log_4 16 + \varepsilon}\right)$ per $\varepsilon \leq \log_4 a - 2 \longleftarrow 2 \geq \log_4 a + \varepsilon$

Titolo nota 17/03/2016

caso 3: $T(n) = \Theta(n^2)$

$a < 16$ regolarità $a\left(\frac{n}{4}\right)^2 \le cn^2$ $\quad a\frac{n^2}{16} \le cn^2$ $\quad c \ge \frac{a}{16}$

$a > 16$ $f(n) = n^2 = O\left(n^{\log_4 a - \varepsilon}\right)$ per $\varepsilon \le \log_4 a - 2$

caso 1: $T(n) = \Theta\left(n^{\log_4 a}\right)$ per A' $\quad a > 16$

$T(n) = \Theta\left(n^{\log_2 7}\right)$ per A

$n^{\log_4 a} \le n^{\log_2 7}$ $\quad \log_4 a \le \log_2 7$ $\quad \log_b a = \frac{\log_c a}{\log_c b}$

$a = 49$

$\frac{\log_2 a}{2 \log_2 4} \le \log_2 7$ $\quad \log_2 a \le 2 \log_2 7 = \log_2 7^2$ $\quad a \le 49$

A' vince A

Titolo nota 17/03/2016

array A di n interi positivi

K intero

determinare se esiste una coppia $i, j : A[i] + A[j] = K$

soluzione banale $\Theta(n^2)$

- genero tutte le coppie; $\binom{n}{2} = \Theta(n^2)$
- guardo se ce n'è una a somma K; $\Theta(1)$

soluzione migliore

- ordino A con MergeSort $\Theta(n \log n)$
- scandisco l'Array sfruttando l'ordinamento $\Theta(n)$

$\Theta(n \log n) + \Theta(n) = \Theta(n \log n)$

Titolo nota 17/03/2016

```
Cerca coppia (A, k);        // input: A ordinato
i = 1; j = n
while (i < j) {
    if (A[i] + A[j] == k) return <i, j>
    else if (A[i] + A[j] < k) i++;
    else j--;
} return <-1, -1>;
```

$O(n)$

$O(n \log n) + O(n \log n)$

Titolo nota 17/03/2016

A: ordinato $\forall i$ vado a fare RICERCA BINARIA $K - A[i]$

```
cercaCoppia2 (A, K)        input A ordinato
for (i = 1, i < n, i++) {
    u = RicercaBinaria(A, K - A[i], 1, n)
    if (u ≠ -1 && u ≠ i)  return <i, u>;
}
return <-1, -1>;
```

$O(\log n)$

Complessità $O(n \log n) + \Theta(n \log n) = \Theta(n \log n)$

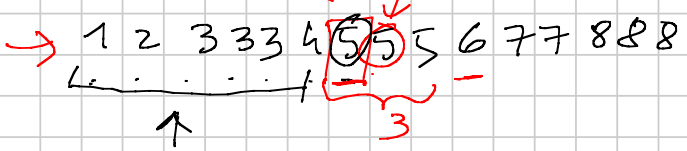
MS

Titolo nota 17/03/2016

A ordinato n interi positivi con ripetizioni
contare le occorrenze di K

occ = numero di occorrenze di K

K=5



<u>banale</u>

```
Conta (A, K):
i = 1; ctr = 0;
while (i ≤ n && A[i] ≤ K) {
    if (A[i] == K)  ctr++;
    i++;
}
return ctr;
```

$\Leftarrow O(n)$

Trovare una soluzione

$O(\log n + \underline{occ}) = O(n)$

RICERCA BINARIA (A, K).

$\Leftarrow \Rightarrow$

Titolo nota 17/03/2016

Ricerca Binaria si può modificare il modo che restituisca (in caso di K ripetuti) la posizione del K più a sinistra

$A[i] : K \begin{cases} < & \text{va a sin.} \\ = & K \quad \text{si ferma} \\ > & \text{va destra} \end{cases}$

$A[i] : K \begin{cases} \leq & \text{sinistra} \\ > & \text{destra} \end{cases}$

Titolo nota 17/03/2016

Input:
array ord A

Output:
posizione in A del K + a sin

RicercaBinariaSx (A, K, sx, dx);

if (sx > dx) return -1;

if (sx == dx)

if (A[sx] == K) return sx;

else return -1;

- c = (sx + dx)/2;

if (K ≤ A[c]) return RicercaBinariaSx (A, K, sx, c);

else return RicercaBinariaSx (A, K, c+1, dx);

$\Theta(\log n)$

k = 4

A

1	2	3	4	5	6	7	8	9	10
1	2	4	4	4	8	8	9	10	11

c = 5 c = 3 c = 2 2,2

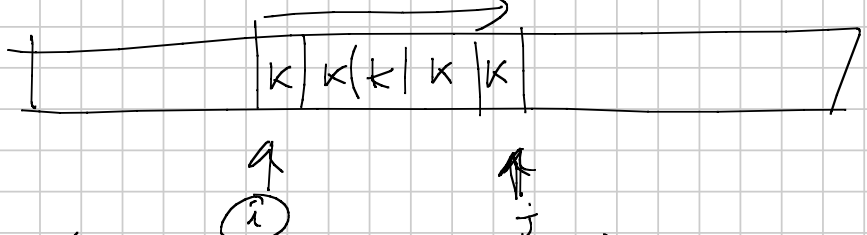
3

Titolo nota 17/03/2016

$O(\log n + occ)$ $O(n)$

1) i = Ricerca binaria sx $\Theta(\log n)$
2) j = Ricerca binaria dx $\Theta(\log n)$
3) $occ = j - i + 1$ $\Theta(1)$

$\Theta(\log n)$



n° occorrenze = $j - i + 1$