Es. 2

Monday, March 9, 2020 10:29 AM

Ricerca Binaria

$a_1 \leq a_2 \ldots \leq a_n$

Divide et Impera

$a[i] \leq k$

Test $a[i]=k$ è rimandato a quando il sottoinsieme in esame ha dimensione pari a 1.

```
RicercaBinaria3 (a, k, sx, dx):
if (sx > dx) return -1;
    if (sx == dx) {
        if (a[sx] == k) return sx;
        else return -1;
    }
cx = (sx+dx)/2;
if (k <= a[cx]) return RicBin(a,k,sx,cx);
    else return RicBin(a,k,cx+1,dx);
```

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

RicBin1 $O(\log n)$

RicBin2 $O(\log n)$

a								
	14	43	76	100	115	290	500	511
	1	2	3	4	5	6	7	8

$k = 76$

sx	dx	cx
1	8	4
1	4	2
3	4	3
3	3	

$k \leq 100$? sì

$k \leq 43$? no

$k \leq 76$? sì

$a[sx] = 76$

È utile quando ci sono elementi ripetuti, restituisce la posizione del k con valore minimo.

1	11	43	54	76	76	76	81
1	2	3	4	5	6	7	8

$K = 46$

sx	dx	cx
1	8	4
5	8	6
5	6	5
5	5	5

$K \leq a[6] = 76$

$K \leq a[5] = 76$

$k = 76$

Ricerca Binaria 2 ? la posizione di K che viene restituita è casuale.

1) Trasformare RicBin3 in modo tale che restituisca la posizione max nel caso di elementi ripetuti.

2) Definire un algoritmo efficiente che dato a di n con valori ripetuti ordinato mi restituisca il n° di el. = K

UGUALI (a, n, K)

algoritmo banale $O(n)$

algoritmo efficiente $O(\log n)$

Problema della Ricerca n elementi, K

Ric. Sequenziale $O(n)$

Ric. Binaria $O(\log n)$

$O(\log \log n)$??

Cerchiamo di individuare un limite inferiore al problema della ricerca

Problema delle 12 monete : pesata

operazione di confronto tra 2 el.

operazione di [illegible]

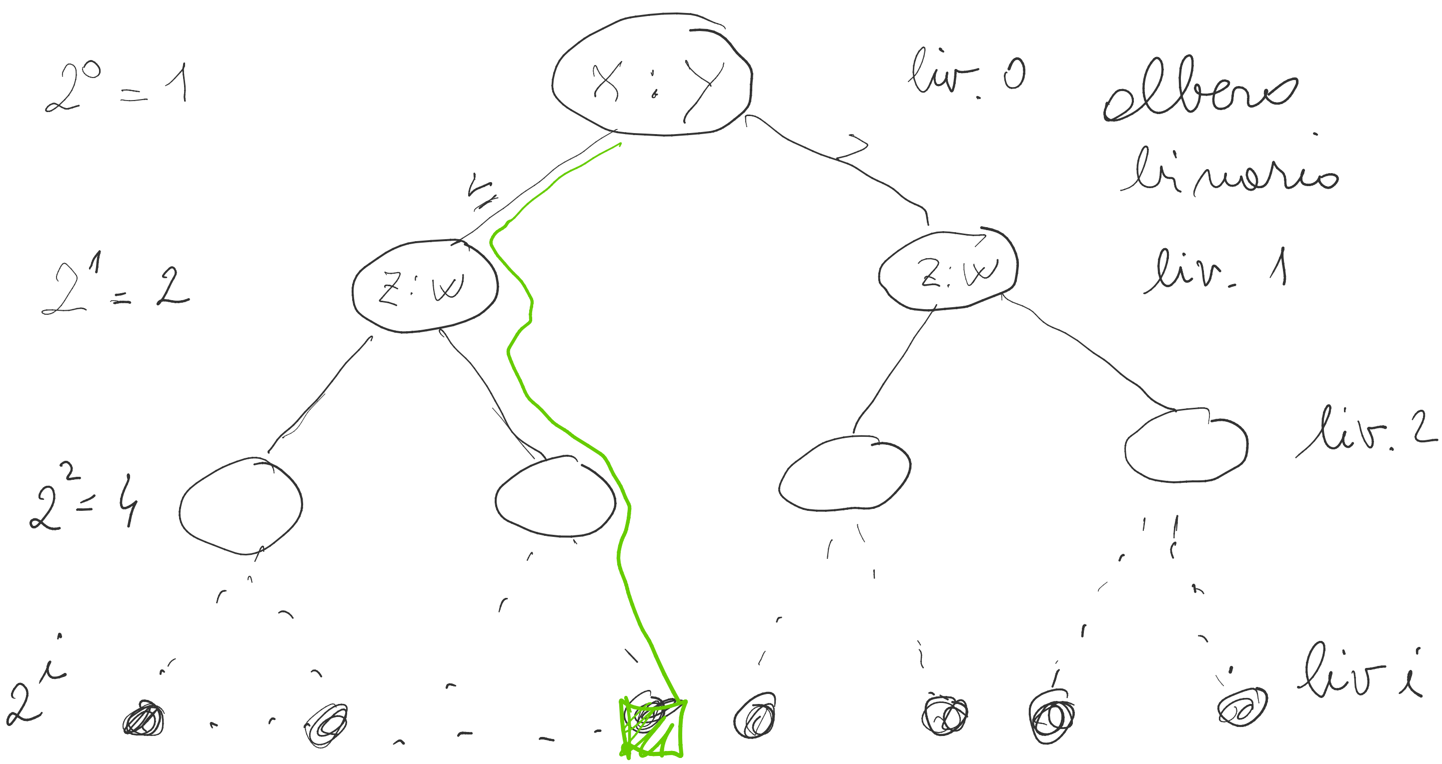
1) <u>array non ordinato</u>

2) <u>array ordinato</u>

1) $\Omega(n)$ → tutti elementi devono essere confrontati $n/2$ sono necessari → $\Omega(n)$

2) si procede in maniera simile al problema delle 12 monete.

Albero di decisione



Quante soluzioni ?

$$\left.\begin{aligned} k &= a[1] \\ k &= a[2] \\ &\vdots \\ k &= a[n] \end{aligned}\right\} \text{ ricerca con successo}$$

$k \notin a$ ricerca senza successo

$$|S| = n+1$$

dobbiamo poter distinguere sull'albero di decisione su $n+1$ possibilità

$$\boxed{2^i \geq |S| = n+1}$$

$$2^i \geq n+1 \qquad \log_2 2^i \geq \log(n+1)$$

$$i \geq \log(n+1)$$

$$\boxed{i} = \Omega(\log n)$$

i è il n° di confronti necessari a trovare k nel caso pessimo.

array ordinato

Ric. Binaria $\quad O(\log n)$ operazioni

limite inferiore $\quad \Omega(\log n)$

poiché i due limiti sono equivalenti in

perché ... ordine di grandezza

Ricerca Binaria è un alg. ottimo

Stabilire limiti inferiori non è un problema semplice.

Problema relativo Merge Sort

1) Scrivere una procedura che faccia il MS per $n > k$ e InsSort per $n \leq k$. $\boxed{n = 2^k}$

2) Valutare la complessità in funzione di n e k.

MergeInsSort (a, sx, dx, k)

3) $IS = 2n^2$ $MS = 64 n \log n$

trovare il massimo valore di n per cui conviene usare IS.

1) MergeInsSort (a, sx, dx, k):

if ((dx - sx + 1) > k) { usa MergeSort

$cx = \frac{sx + dx}{2}$;

MergeInsSort (a, sx, cx, k);

$\qquad$ 2

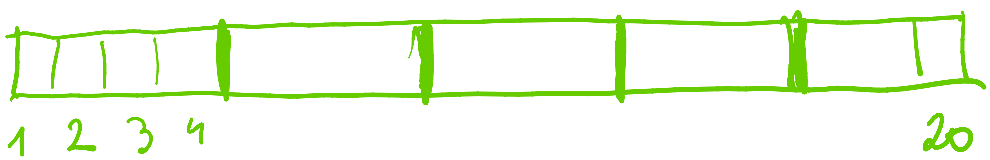
Merge InsSort (a, sx, cx, k);

MergeInsSort (a, cx+1, dx, k);

Merge (a, sx, dx, cx);

} else $\qquad$ use InsSort : $(dx - sx + 1) \le k$

InsertionSort (a, sx, dx);



$K = 4$

$\frac{n}{k} = \frac{20}{4} = 5$ sottoinsiemi di dim 4

2) Complessità

$$T(n) = \begin{cases} 2T\left(\frac{n}{2}\right) + \Theta(n) & n > K \\ O(n^2) = O(k^2) & n \le K \end{cases}$$

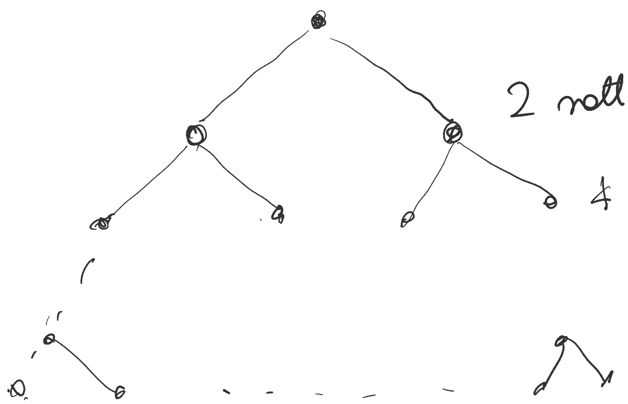
1) $n \le K$ $\qquad O(k^2)$ tempo necessario per ordinare un sottoinsieme di k elementi

in totale il tempo di InsSort

$$O(k^2) \cdot \frac{n}{\cancel{k}} = \boxed{O(n \cdot k)}$$

2) fase merge

2) fase merge



2 sott array di $\frac{n}{2} \Rightarrow n$ op. merge

4 sottoarray di $\frac{n}{4} \Rightarrow n$ op. merge

$i \rightarrow$ 2^i sottoarray di $\frac{n}{2^i} \Rightarrow n$. op. merge

$\frac{n}{2^i} = K \qquad 2^i = \frac{n}{K} \qquad i = \log(n/K)$

$\Theta(n \cdot \log(n/K))$ fase merge

COSTO TOTALE $\Theta(n \log(n/K) + n \cdot K)$

3) $IS = 2n^2 \qquad MS = 64\, n \log n$

max val di n per cui conviene usare IS.

$$2n^2 \leq 64\, n \log n$$
$$n \leq 32 \log n$$

proviamo con

$n = 2^7 = 128$

$128 \leq 32 \cdot 7 = 224$ conviene IS

$n = 2^8 = 256$

$n = 2^8 = 256$

$2^8 \leq 2^5 \cdot 8 = 2^5 \cdot 2^3 = 2^8$ IS = MS

per $n = 257$ conviene MS

$n = 256$ è il max val per cui

conviene IS.