Lezione 10 marzo 2020
Tuesday, March 10, 2020 3:34 PM

SEARCH & SORTING

ORDINAMENTO

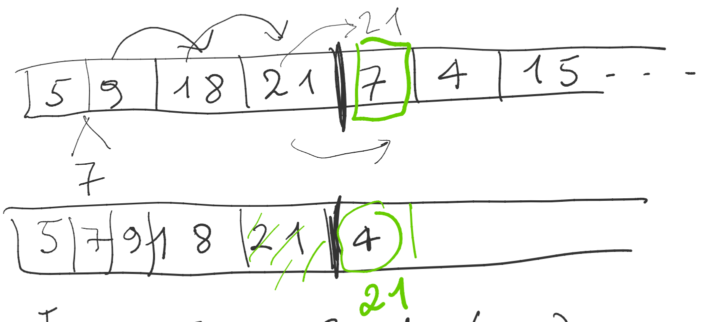
input : $a_1, \dots, a_n$

output: $a_{i_1} \le a_{i_2} \le \dots \le a_{i_n}$

permutazione ordinata

Insertion Sort

inserisce l'elemento corrente nel sottoinsieme già ordinato.



prox = 7

prox = 4

Insertion_Sort (a):

```
for (i = 1; i < n; i++) {
    prox = a[i];
    j = i;
    while (j>0) && (a[j-1] > prox) {
        a[j] = a[j-1];      → shift a destra
        j--;
    }
    a[j] = prox;
}
```

$O(n^2)$ $O(n)$

Insertion Sort = $O(n^2)$

Insertion Sort = $O(n^2)$

$\Theta(n^2)$? NO

3	5	9	12	15	18	20	25

↑ prox

∀ ogni el. si esegue un confronto

caso ottimo : $\Theta(n)$

caso pessimo : (array in input è ordinato in ordine non crescente)

10	7	5	4	3	2	1

7	10	

n° confronti nel caso pessimo =

$$1+2+3+\ldots+n-1 = \frac{n \cdot n-1}{2} = \Theta(n^2)$$

IS è $O(n^2)$

Correttezza

proprietà invariante che si dimostra vera

- all'inizio
- prima e dopo ogni iterazione del ciclo
- la verità della proprietà alla fine dell'esec. dell'alg. garantisce il risultato

all'iterazione i :

a[0..i-1] è ordinato

$a[0..i-1]$ è ordinato

- $a[0]$ ovviamente ordinato
- $a[0..i-1]$ è ordinato prima dell'i-esima iterazione
- $a[0..i]$ è ordinato

- $a[0..n-1]$ è ordinato □

Selection_Sort (a):

idea: ad ogni iterazione si cerca il minimo dell'insieme e si porta all'inizio.

n iterazioni

1: min di n elementi $n-1$
2: min di $n-1$ " $n-2$
⋮
$n-1$: min di 2 elementi   1

In tutti i casi il n° confronti =
$$1+2+\dots(n-2)+(n-1) = \Theta(n^2) \text{ confronti}$$

Selection Sort è $\Theta(n^2)$

Esercizio: scrivere la procedura per Sel Sort
analisi di correttezza

operazione base: confronto tra elementi

Paradigma di costruzione di alg.

. Divide et Impera

. Divide and Conquer

- Dividi il problema da risolvere in sottoproblemi più piccoli
- Risolvi risolvi i sottoproblemi ricorsivamente o direttamente se abbastanza piccoli
- Combina trova la soluzione al problema iniziale combinando le sottosoluzioni ai sottoproblemi parziali

Algoritmo MergeSort

Dividi: dividi l'insieme in 2 sottoinsiemi di dim. $n/2$

Risolvi: se il sottoinsieme è di un elemento è già ordinato altrimenti ordina ricorsivamente.

Combina: fai la fusione (Merge) di 2 sottoinsiemi ordinati di $n/2$ el. ciascuno per ottenere l'insieme ordinato di n elementi

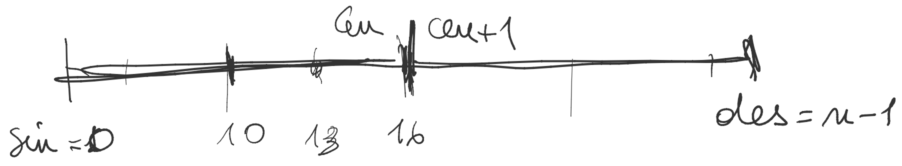
MergeSort (a, sin, des): MS(0,3)

if (sin < des) {

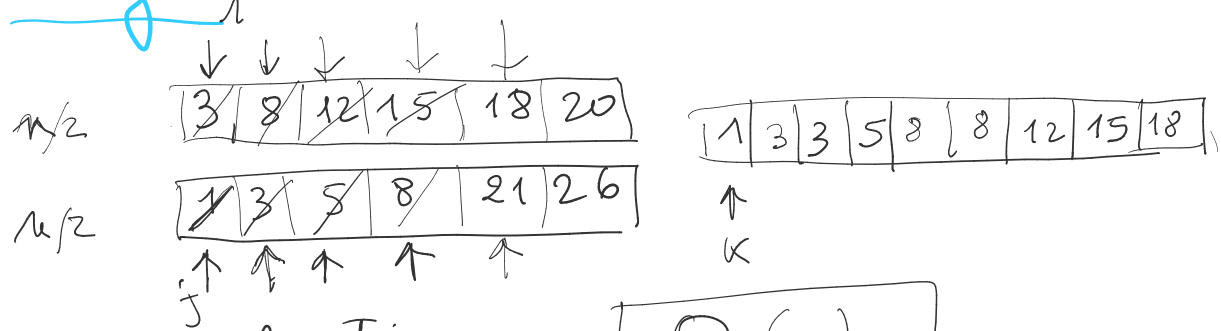
$T(n) = \begin{cases} \Theta(1) & n = 1 \\ \ldots \end{cases}$

```
if (sin < des) {
    cen = (sin + des) / 2;
    MergeSort(a, sin, cen);
    MergeSort(a, cen+1, des);
    Merge(a, sin, cen, des);
}
```

$$T(n) \simeq \begin{cases} \Theta(1) & n=1 \\ 2T\left(\frac{n}{2}\right)+\Theta(n) & \end{cases}$$



Merge



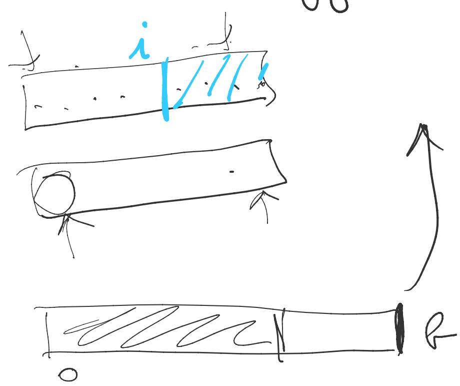
$n-1$ confronti

$\Theta(n)$ spostamenti

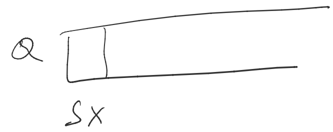
$\boxed{\Theta(n)}$

```
Merge (a, sx, cx, dx):
i = sx; j = cx+1; k = 0;
while (i ≤ cx) && (j ≤ dx) {
    if a[i] ≤ a[j] {
        b[k] = a[i];
        i++;
    }
    else { b[k] = a[j];
        j++; }
    k++;
} for ( ; i ≤ cx; i++; k++) b[k] = a[i];
```

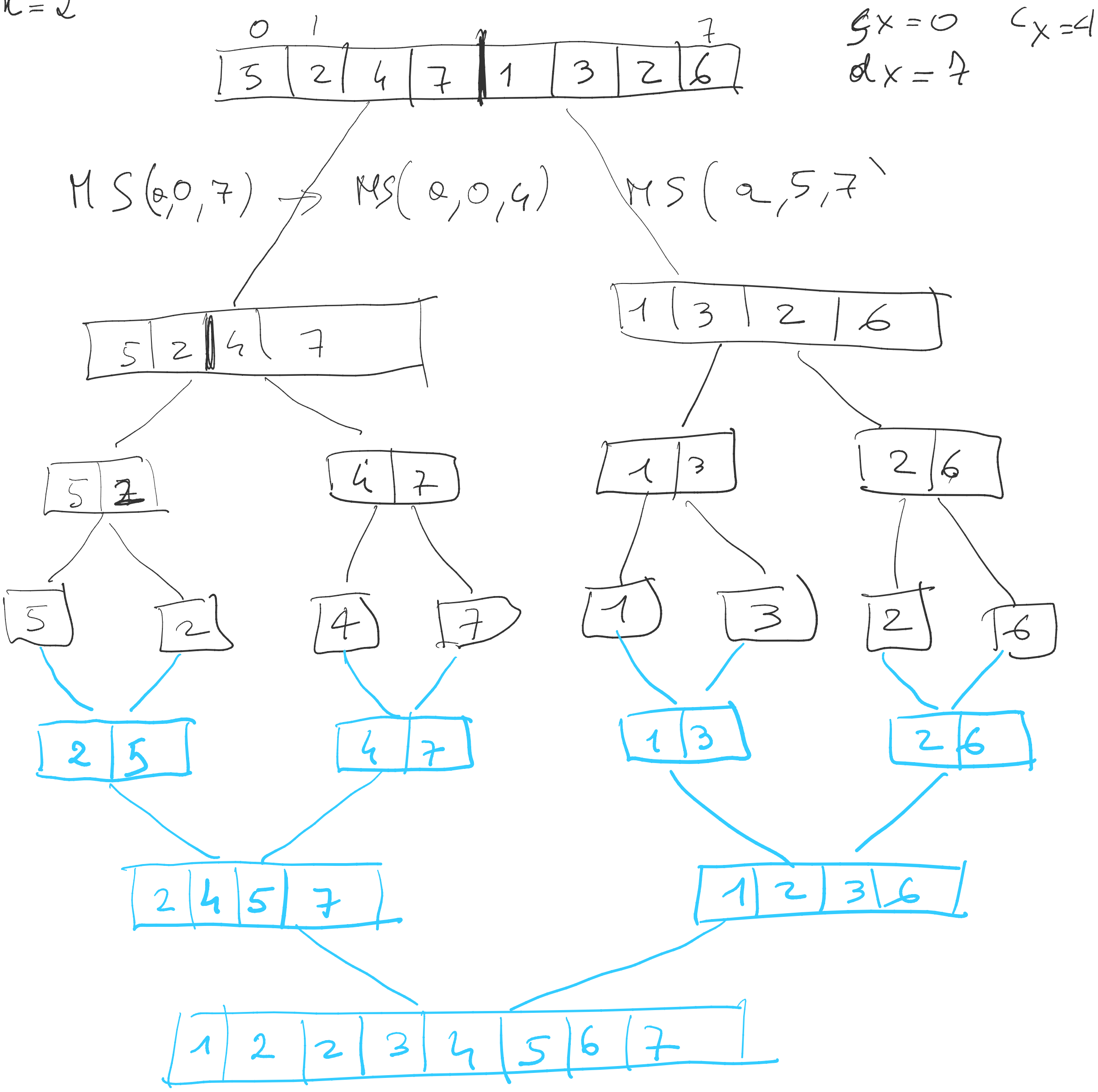
output:
a [sx.. dx] ordinato
b: array di appoggio



```
          k++;
} for ( ; i ≤ cx ; i++; k++)      b[k] = a[i];
  for ( ; j ≤ dx ; j++ ; k++)     b[k] = a[j];
  for (i = sx; i ≤ dx ; i++)
        a[i] = b[i-sx];
```



$n = 2^k$



$$T(n) = \begin{cases} \Theta(1) & \text{per } n = 1 \\ 2\,T\left(\frac{n}{2}\right) + \Theta(n) & \text{per } n > 1 \end{cases}$$

2 chiamate ricorsive di Merge Sort

tempo di Merge

ricorsive di
Merge Sort
su $n/2$ elementi

$n = 2^i$

$$T\left(\frac{n}{2}\right) = 2\,T\left(\frac{n}{4}\right) + \Theta\left(\frac{n}{2}\right)$$

$$T(n) = 2\left(2\,T\left(\frac{n}{4}\right) + \frac{n}{2}\right) + n \qquad T\left(\frac{n}{4}\right) = 2\,T\left(\frac{n}{8}\right) + \frac{n}{4}$$

$$T(n) = 2\left(2\left(2\,T\left(\frac{n}{8}\right) + \frac{n}{4}\right) + \frac{n}{2}\right) + n =$$

$$2\left(2\left(2\,T\left(\frac{n}{2^3}\right) + \frac{n}{2^2}\right) + \frac{n}{2^1} + \frac{n}{2^0}\right.$$

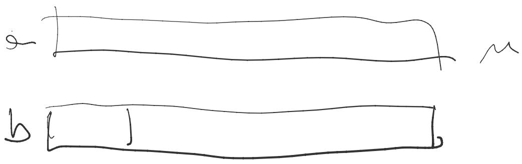
$$T(n) = 2\left(2 \cdots \left(2\,T\left(\frac{n}{2^i}\right) + \frac{n}{2^{i-1}}\right) + \frac{n}{2^{i-2}}\right) + \cdots\Big) + n$$

$$= 2^i\,T(1) + \underbrace{\cancel{2^{i-1}}\,\frac{n}{\cancel{2^{i-1}}} + \cancel{2^{i-2}}\cdot\frac{n}{\cancel{2^{i-2}}} + \cdots + n}$$

$$= n + i \cdot n$$

$$= n + n\log n = \boxed{\Theta(n \log n)} \qquad 2^i = n \qquad i = \log_2 n$$

MS è $\Theta(n \log n)$ tempo
$\Theta(n)$ spazio



spazio aggiuntivo
l'array b di n el.

Teorema principale dell'eq. di ricorrenza

$$\Theta(n^2) \longrightarrow O(n^2) \longrightarrow \Theta(n \log n)$$

Sel Sort — In Sort — Merge Sort

Possiamo trovare un algoritmo ancora

Possiamo trovare un algoritmo ancora migliore?

- Determinare un limite al problema dell'ordinamento.

Esercizio

1) Definire una procedura di ordinamento che faccia il MERGESORT se $n > K$ e l'IS se $n \leq K$.

2) Calcolare la complessità asintotica e discutere per quali valori di K è conveniente

3) Come si fa a stabilire in pratica il valore di K.

MERGEINSSORT (a, sx, dx, K):