Selection sort

1

SS

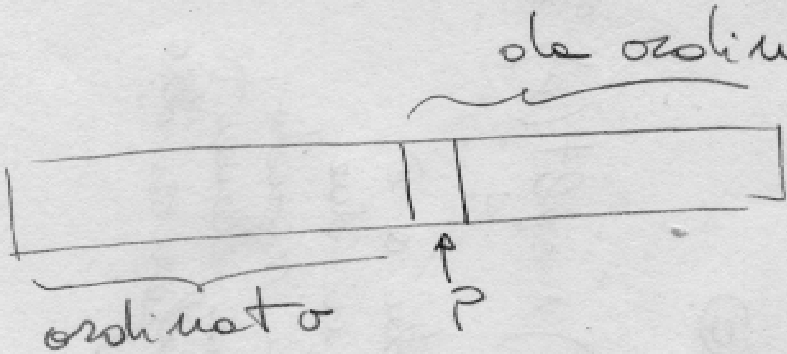
Idea: 1) array diviso in due parti
- a[0] ... a[p-1] ordinata
- a[p] .. a[a.len] da ordinare

2) si cerca la posizione del minimo tra p e a.len-1

3) si scambia il minimo con a[p]

4) si incrementa p, siccome la parte ordinata è appena stata estesa

5) si ricomincia da 2.



2

Sort

```
      p = 0;
(a) → while (p < a.len) {
(b) →     m = minimo(a, p)
(c) →     scambio(a, m, p)
(d) →     p = p+1          ← (e)
      }
```

In (b) vale $sort(p) \wedge split(p) \overset{def.}{\equiv} \underbrace{\left(\forall k.\ 0 \le k < p \Rightarrow a[k] \le a[k+1]\right)}_{}^{sort(p)} \wedge$

$$\underbrace{\left(\forall k, j.\ 0 \le k < p \wedge p \le j < a.len \Rightarrow a[k] \le a[j]\right)}_{split(p)}$$

indica che a è diviso in due parti: la seconda contiene elementi non inferiori alla prima

In (c) sappiamo dove si trova il minimo della parte non ordinata.

$$\forall i.\ p \leq i < a.len \Rightarrow a[m] \leq a[i] \stackrel{def}{\equiv} min(m, p)$$

In (d) sappiamo che il minimo della parte ordinata è in $a[p]$;

$$\forall i\ \ p \leq i < a.len \Rightarrow a[p] \leq a[i] \stackrel{def}{\equiv} min(p)$$

Possiamo dedurre quanto segue:

$$sort(p) \wedge split(p) \wedge min(p) \Rightarrow sort(p+1) \wedge split(p+1)$$

ordinata fino a p

la parte ordinata ha elementi non superiori a quelli da ordinare

$a[p]$ non ha elementi superiori a sinistra e non ha elementi inferiori a destra

In e, avendo incrementato p abbiamo $sort(p) \wedge split(p)$

Infine in ⓐ valgono sort(a) ∧ split(a) per il solito motivo

$$\left(\forall k.\ \underbrace{\underbrace{0\le k<a}_{F} \Rightarrow \underbrace{a[k]\le a[k+1]}_{F}}_{V}\right) \wedge \left(\forall k,j.\ \underbrace{\overbrace{\overbrace{0\le k<p}^{F}\ e\ p\le j<a}^{F}.\ \mathrm{ln} \Rightarrow \underbrace{a[k]\le a[1]}_{F}}_{V}\right)$$

ssort.

Proprietà

- La ricerca del minimo elemento richiede di percorrere tutta la parte non ordinata che decresce di un elemento ad ogni ciclo.

  $$(n-1)+(n-2)+\dots+2+1 = \frac{n(n-1)}{2} \text{ confronti}$$

  e $n-1$ swap
- stabile nel senso che non scambia elementi con stesso valore
- Genera Heapsort che ottimizza ricerca del minimo