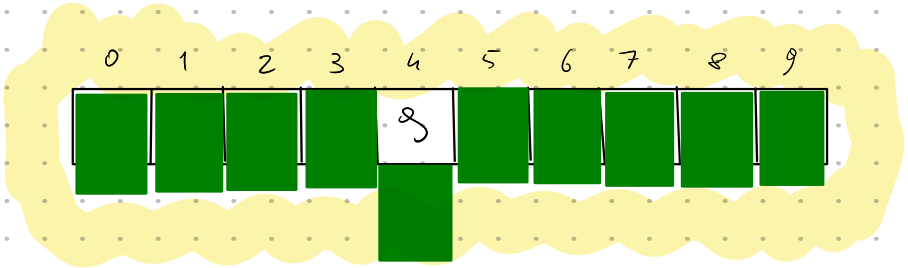
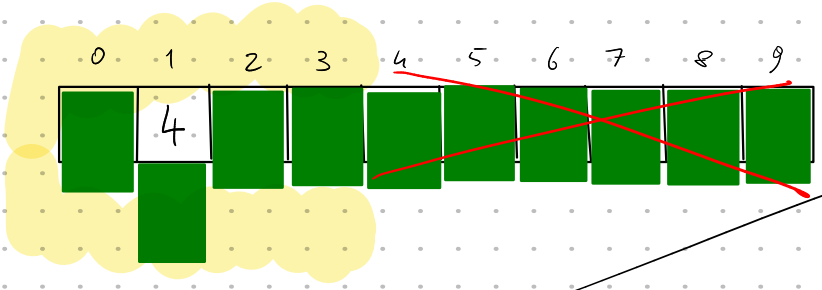
RICERCA BINARIA DEL NUMERO 5

PASSO 0

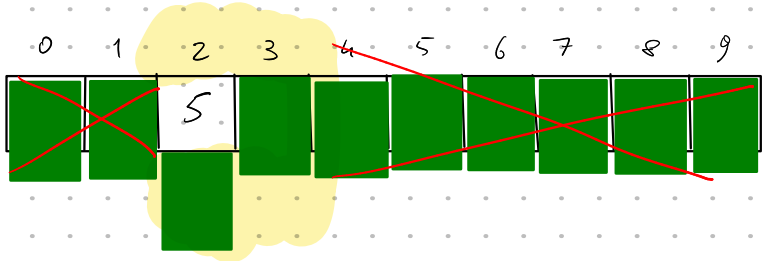


PARTE DELLA LISTA DOVE PUO' TROVARSI IL 5

PASSO 1

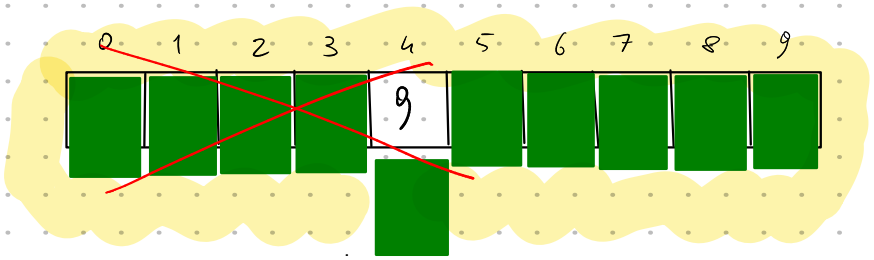


PASSO 2

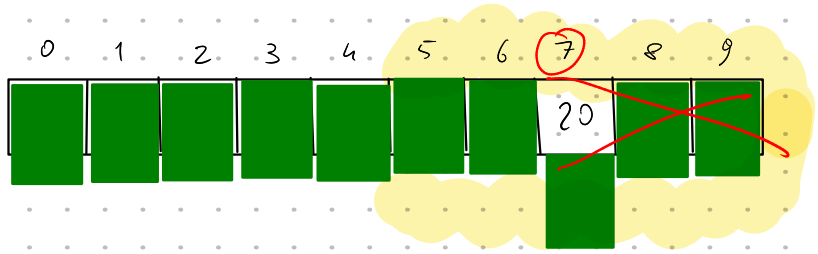


RICERCA BINARIA DEL NUMERO 10

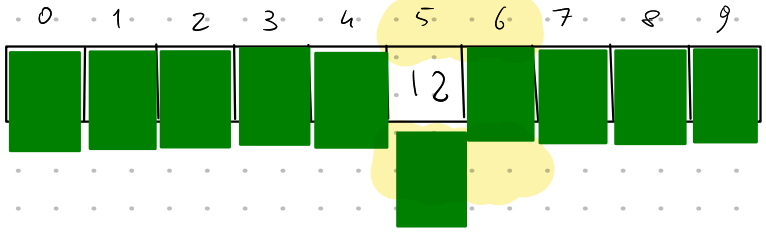
PASSO 0

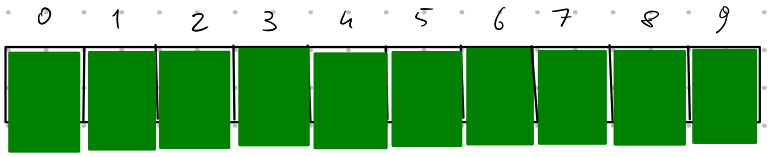


PASSO 1



PASSO 2

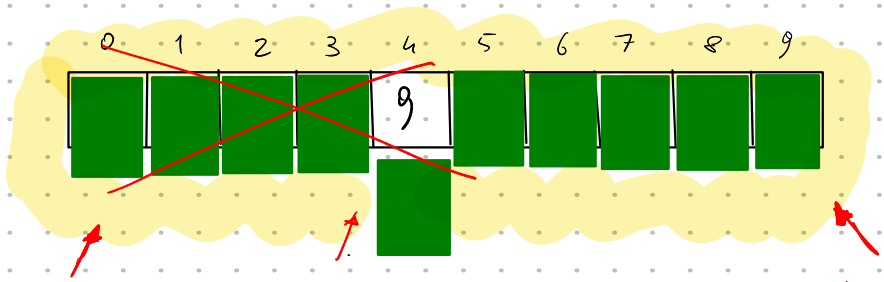




NON C'È PIÙ NIENTE DA CONTROLLARE, IL VALORE 10 NON SI TROVA NELLA LISTA

RICERCA BINARIA DEL NUMERO 10

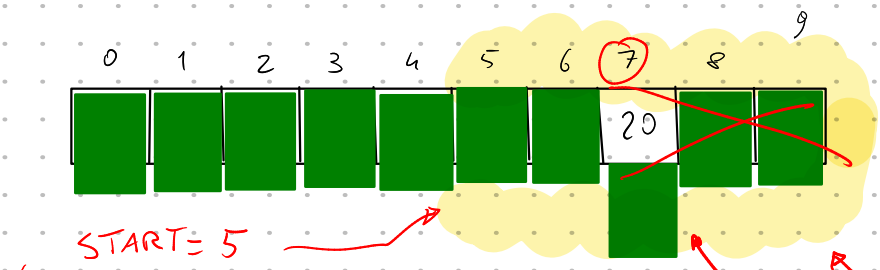
PASSO 0



START = 0 MIDDLE = 4 END = 9 = len(l) − 1

LE VARIABILI START E END INDICANO IL PRIMO E ULTIMO ELEMENTO DELLA ZONA GIALLA

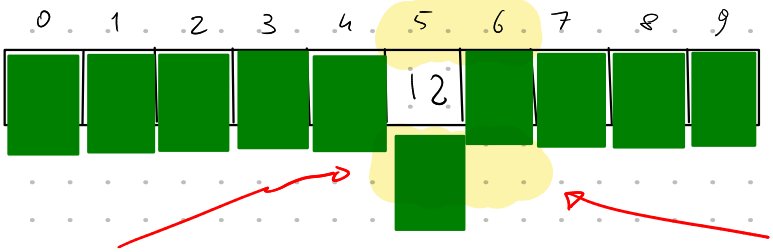
PASSO 1



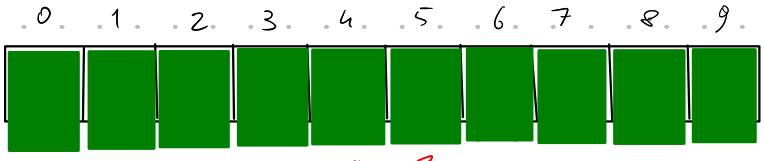
START = 5 (IL VECCHIO MIDDLE + 1) MIDDLE = 7 END = 9

LA VARIABILE MIDDLE CONTIENE LA POSIZIONE DA CONTROLLARE, ED È LA PARTE INTERA DI $\frac{START + END}{2}$

PASSO 2



START = 5 MIDDLE = 5 END = 6 (IL VECCHIO MIDDLE − 1)



START = 5 END = 4 (IL VECCHIO MIDDLE − 1)

NON C'È PIÙ NESSUNA PARTE DA CONTROLLARE, INFATTI È START > END

ANALISI DELLA COMPLESSITÀ NEL CASO PESSIMO

RICERCA LINEARE

Chiamiamo n la lunghezza della lista in input.

Il caso pessimo è quello in cui l'elemento che cerchiamo non esiste nella lista.

Domanda: quanti accessi facciamo alla lista prima di rispondere?

Risposta: n

RICERCA BINARIA: vogliamo rispondere alle stesse domande.

S_i è il numero di elementi nella lista che possono contenere il valore che cerco al passo i, ovvero il numero di elementi marcati in giallo al passo i.

$S_0 = n$ all'inizio il valore da cercare può essere ovunque

$S_1 \leq n/2 = \frac{n}{2^1}$

$S_2 \leq S_1/2 = n/4 = \frac{n}{2^2}$

$S_3 \leq S_2/2 = n/8 = \frac{n}{2^3}$

$S_i \leq S_{i-1}/2 = \frac{n}{2^i}$

dopo ogni passo, la dimensione dell'area gialla si restringe a meno della metà di quella al passo precedente

Quando S_i diventa minore di 1, sicuramente l'algoritmo termina, perché non c'è più nulla da controllare!

Ma dopo quanti passi (ovvero, per quale valore di i) accade questo?

Dobbiamo risolvere l'equazione $S_i < 1$

$\frac{n}{2^i} < 1$ equazione in $\boxed{i}$

$\frac{n}{2^i} \cdot 2^i < 2^i$ moltiplico ambo i membri per 2^i

$\log_2 n < \log_2 2^i$ applico il $\log_2$ ad ambo i membri

$\log_2 n < i$

Appena i supera $\log_2 n$, la procedura si interrompe di sicuro (se non si è già interrotta prima). Quindi il numero massimo di iterazioni è $\lfloor \log_2 n \rfloor + 1$.

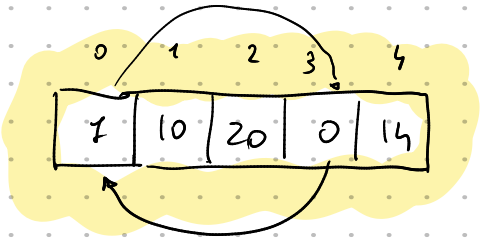
$\lfloor \log_2 n \rfloor + 1$ è molto più piccolo di n!

- Se $n = 400.000$ allora $\lfloor \log_2 n \rfloor + 1 = 18$
- Se $n = 400$ miliardi allora $\lfloor \log_2 n \rfloor + 1 = 38$

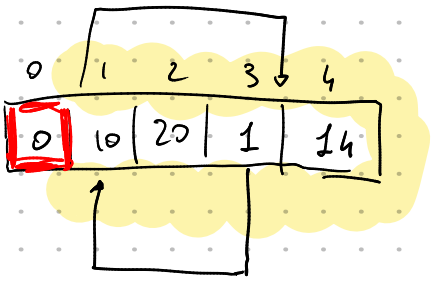
Per cercare con la ricerca binaria un elemento in una lista di 400 miliardi di elementi, bastano 38 tentativi!

ORDINAMENTO PER SELEZIONE

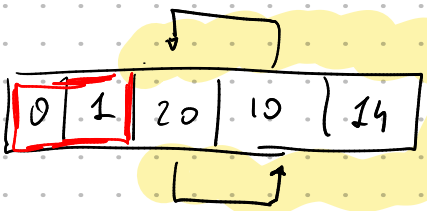
PASSO 0)



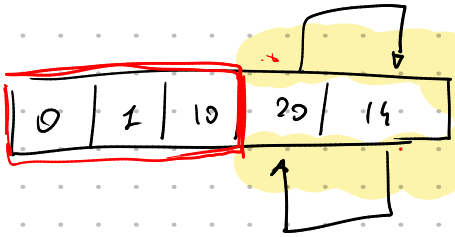
PASSO 1)



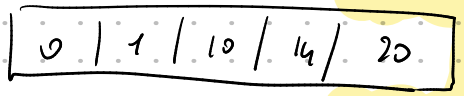
PASSO 2)



PASSO 3)



PASSO 4)



ALCUNI ALGORITMI DI ORDINAMENTO

- Scambiare elementi adiacenti

  BUBBLE SORT

- ripetutamente selezionare l'elemento massimo / minimo e metterlo al posto giusto

  SELECTION SORT

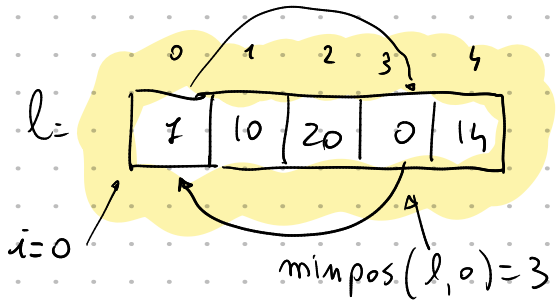
In giallo la parte della lista ancora da ordinare, in rosso quella già ordinata.

Ad ogni passo trovo il minimo della parte gialla e lo scambio con l'elemento che sta nella prima posizione della parte gialla.

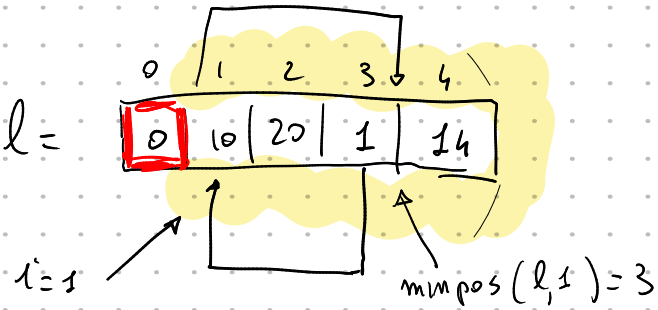
Quando rimane un solo elemento nella parte gialla posso fermarmi perché non c'è nessun ulteriore spostamento da fare.

ORDINAMENTO PER SELEZIONE

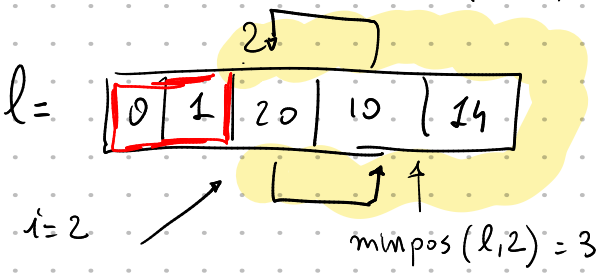
PASSO 0)



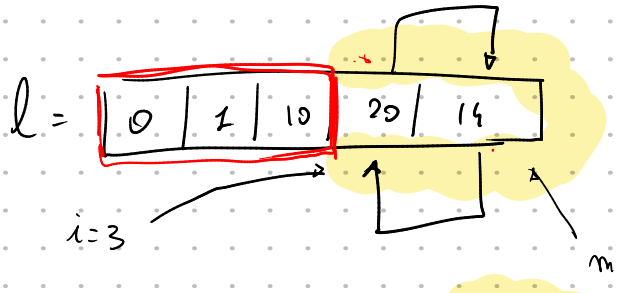
PASSO 1)



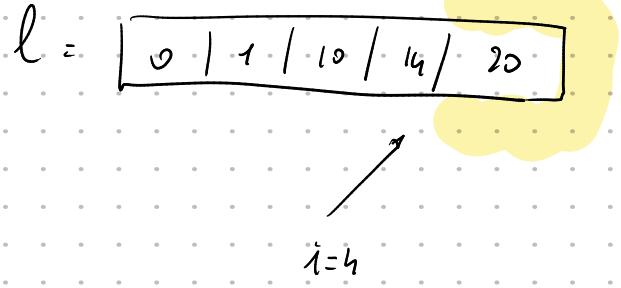
PASSO 2)



PASSO 3)



PASSO 4)



Per implementare l'algoritmo serve:

- una variabile i che contiene l'indice in cui inizia la zona gialla
- una funzione che restituisce la posizione in cui si trova il valore minimo della zona gialla o, in generale, una funzione $minpos(l, j)$ che restituisce la posizione del valore minimo tra $l[j]$, $l[j+1]$, $l[j+2]$, ... fino all'ultima posizione di l.

Posso fermarmi quando i punta all'ultimo elemento di l, ovvero $i = len(l) - 1$.