```
while (k <= A.length && A[k].l ≤ m) {
                        m ← max (m, A[k].r)
                        k ← k+1
}
A'[j].r ← m
i ← k+1
j ← j+1
}
}
```

No Altrimenti sarei in grado in meno di $O(n \log n)$ nel caso peggiore di decidere se un vettore contiene ripetizioni

⇓

Non si può
Conseguenza $\Omega(n \log n)$
caso peggiore ordinamento

10/12 Lab

Relazione 10/15 pag
Non serve spiegazione algoritmi
Bisogna andare a descrivere il mostro contributo, osservazioni interessanti, grafici...

Algoritmi di selezione → input array e indice k restituendo l'elemento a[i] tale che è il k esimo elemento nel vettore ordinato

Risolvere il problema in 3 modi → quick heap e median of median select

1) Implementare 3 algoritmi che siano corretti formalmente e con complessità data

Quick Select: nel caso medio lineare
Heap Select: nel caso medio $(n + k \log k)$

Gestisco k costante uguale a 100 a seconda della rilevanza e per ognuno genero un grafico interessante $(n + 100 \log 100)$..

Median of medians select: complessità lineare data l'ipotesi uniforme

2) Stima tempi medi di esecuzione
ci saranno grafici in cui vengono stimati i tempi di esecuzione dipendente da array length

grafici confrontativi in scala semplice, ma anche su scala logaritmica per far vedere cosa accade per valori molto grandi e molto piccoli, applicare logaritmi sia su ordinate e su ascisse

Molto interessante è studiare la stima dei tempi per n fissato e clock con k variante si può analizzare

Per misurare il tempo di esecuzione una opzione (brutta) è fare una media di tante esecuzioni (tipo 100). Ma non è molto sicuro

Problema
- x incognita
- R risoluzione
- $\tilde{x}$ una misurazione di x $\quad x-R \leq \tilde{x} \leq x+R$
- E errore assoluto $\quad E = |x-\tilde{x}|$
- e errore relativo $\quad e = E/x$
- e_{max} errore relativo massimo voluto (errore relativo al più accettato)
  $e_{max} = 0{,}001 = 0{,}1\%$

$$e = \frac{E}{x} = \left|\frac{x-\tilde{x}}{x}\right| \leq \frac{R}{x}$$ (errore relativo se misurassi xtilde)

$X_m = x \cdot m$ ripeto m volte il mio oggetto x e tutti i parametri descritti sopra ma dipendente da m

$$E_m = \frac{e_m}{x_m} = \frac{|X_m - \tilde{X}_m|}{x_m} \leq \frac{R}{x_m} = \frac{R}{x \cdot m} \overset{!!}{\leq} e_{max} \quad \text{quando } x_m \geq \frac{R}{e_{max}}$$

$$\frac{R}{x_m} \leq e_{max} \Leftarrow x_m \geq \frac{R}{e_{max}} \Leftarrow \tilde{X}_m \geq R + \frac{R}{e_{max}}$$

$x_m \geq \tilde{x}_m - R$

(Per avere un valore misurato di

```
do {
    input = generateInput(n)
    execute Alg (input)
    now = clock()
    n = n+1
} while( now_start < R + R/emax )
```

potrei trovare un tempo medio per generare l'input con un ciclo che mi genera $\tilde{y}$ come stima tempo medio di generateInput() ottenendo $\tilde{z} = \tilde{x} - \tilde{y}$ se la generazione dell'input è lineare nel tempo con $\tilde{z}$ tempo dell'algoritmo

PSEUDO CODICE

```
for( i = 1 to [2 ... 100]) {
        ỹ = goodMeasure(inizializzazione)
        x̃ = algoritmo(inizializzazione)
        z̃ = x̃ - ỹ
        array[i] = z̃
    }
}
```

Per ottenere dati più robusti e calcolare varianze di essi

$n = 10 \ldots\ldots \quad 100000$

se generate l'input prima
bla bla bla
Il Puppis dice di essere
(l'originale un po' si nota)
= DavideCap...

Per generare gli input, è molto meglio generare 1000 input già pronti all'uso e nel momento in cui misuro il tempo medio, uso tali input già generati

Per quanto riguarda il c, bisogna cercare di allocare tutto subito e deallocare alla fine in quanto anche l'allocazione prende tempo

Per il for che fa in modo di far variare n nel tempo, non ha senso variare n nell'ordine dell'unità, bensì si cerca conversione di tipo geometrico (progressione geometrica)

Come fare? $N[j]$ con j da 1 a 1000 che segue una distribuzione esponenziale $\rightarrow A \cdot B^j$ e fare in modo tale che $n(1) = n_{min}$ e

$n(1000) = n_{max}$

$A \cdot B^1 = 100 = A = \frac{100}{B}$ $\quad A \cdot B^1 = n_{min}$ e $A \cdot B^{1000} = n_{max}$ e in questo

maniera ottengo i risultati desiderati

$B^{1000-1} = N_{max} / N_{min}$ quindi $\log B = \frac{\log N_{max} - \log N_{min}}{999}$

e $B = \exp \frac{(\log N_{max} - \log N_{min})}{999}$ $\quad$ e $\quad A = \frac{N_{min}}{B}$

Quindi ottengo $n =$