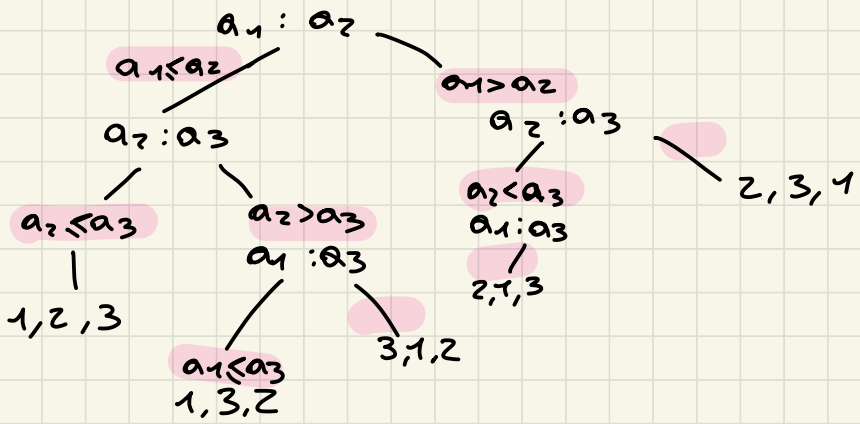
In alcune condizioni riguardanti gli input anche più veloci del quick sort

Algoritmi con confronti

a_1, a_2, a_3



con n numeri

$n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 1 = n!$

non posso farlo solo con n domande

devo farne almeno $2^k \geq n!$ domande

$k \geq \lg_2 n!$ $\quad n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \left(1+\Theta\left(\frac{1}{n}\right)\right)$ $\quad n! \approx \left(\frac{n}{e}\right)^n$

$k \geq \lg_2 \left(\frac{n}{e}\right)^n = n\lg_2 n - n\lg_2 e = \Theta(n \lg n)$

$k = \Omega(n \lg n)$ ← come merge o quick

ordinare in tempo $\Theta(n+k)$

k → dim del max valore in input

Cifre possibili: 1 - 9

A = 7, 1_a, 8, 1_b, 2, 1_c, 5, 4_b, 2, 4_b, 3

1 2 3 4 5 6 7 8 9 10 11

C =

3	5	6	8	9	9	10	11	11
1	2	3	4	5	6	7	8	9

$h = (n) + (n-1)$

① azzero C → $\Theta(k)$

② scansione di A contando in C incremento la corrispondente posizione $\Theta(n)$

B =

1_a	1_b	1_c	2	2	3	4_a	4_b	5	7	8
1	2	3	4	5	6	7	8	9	10	11

③ scambio gli elementi in C in modo da identificare la pos $\Theta(k)$

④ scansione di A al contrario, posizionamento in B (decremento in C quando tocco) $\Theta(n)$

$$T(n) = \Theta(2n + 2k) = \Theta(n + k)$$

```
void countingSort (A[])

    for (i=1 to k){
        C[i] = 0
    }                                   Θ(k)
    for (i=1 to n){
        p = A[i]
        C[p]++                          Θ(n)
    }
    for (i=2 to k){
        C[i] = C[i] + C[i-1]            Θ(k)
    }
    for (i=n down to 1){
        p = A[i]
        B[C[p]] = A[i]
        C[P]--
    }
}
```

DIR
COW
DOG
DIG
DAR

→ COW / DIR / DOG / DIG / DAR

→ COW / DAR / DIR / DIG / DOG

→ COW / DAR / DIG / DIR / DOG

poco efficente, meglio ordinare rispetto l'ultima

DIR
COW
DOG
DIG
DAR

→ DOG / DIG / CAL / DIR / DAR / COW

sono già ordinati, sono uguali dalla prima alla penultima

→ CAL / DAR / DIG / DIR / DOG / COW

ordinati bene

→ CAL / COW / DAR / DIG / DIR / DOG

finito di ordinare

Devo usare un algoritmo di ordinamento stabile

```
RadixSort (A[]){          campi
    for i=0 to K
        sort(A[],i) /* stabile
                       i è il campo meno significativo */
}
}
```

Statistiche d'ordine

min $\rightarrow \Theta(n)$ max $\rightarrow \Theta(n)$

? $\rightarrow n$? $\rightarrow n$

per entrambi

? $2n$

$\frac{n}{2} + \frac{n}{2} + \frac{n}{2} = \frac{3}{2}n$

p G

min max

```
int staOrd ( A[], i, f, iesima ){
    if i<f{
        Q = partition(A[], i, f, ) ← quella del quickSort
        if iesima<Q
            r= statOrd (A , i, q , iesima)
        else
            r= statOrd (A , q+1 , f , iesima-Q )
        return r
    }
    else{
        return i
    }
```

$t(n) = T(\frac{n}{2}) + \Theta(n) = \Theta(n)$

$a=1 \quad b=2 \quad n^{\log_b a} = n^{\log_2 1} = n^0 = 1$