Quick Sort

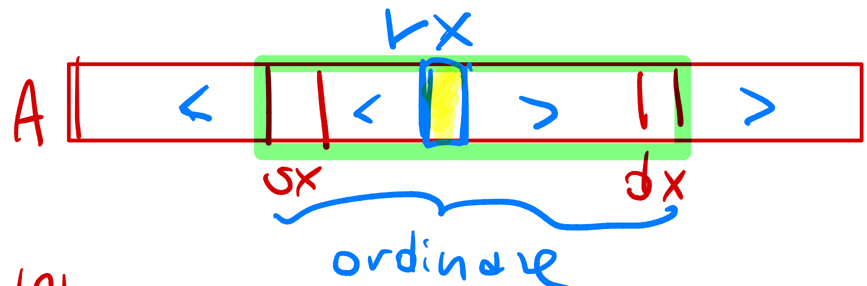
sx dx

```
QuickSort( a, sinistra, destra ):
    ⟨pre: 0 ≤ sinistra, destra ≤ n − 1⟩
  IF (sinistra < destra) {
    scegli pivot nell'intervallo [sinistra...destra];
    rango = Distribuzione( a, sinistra, pivot, destra );
    QuickSort( a, sinistra, rango-1 );
    QuickSort( a, rango+1, destra );
  }
```

$rx = \boxed{\text{rango}}$

pivot = A[rx]

- natura ricorsiva
- $\forall x \in A[0 .. sx-1]:$ $x < y \in A[sx .. dx]$, $\forall y$
- $\forall x \in A[dx+1 .. n-1]:$ $x > y \in A[sx .. dx]$, $\forall y$, $n = |A|$



Esempio

$$A = [4, \overset{1}{\boxed{5}}, 3, 7, 2]$$

Distribuzione (Partition)

$$\begin{array}{ccccc} sx & & & rx & dx \\ 4, & 3, & 2, & \boxed{5}, & 7 \\ & & & 3 & \end{array}$$

ordine qualunque

< 5

> 5

STOP

$$\begin{array}{ccc} sx & & \\ 4, & 3, & 2 \end{array}$$

distribuzione

$rx - 1$

$[3, 2, \boxed{4}]$

STOP

$[3, 2]$

$[2, 3]$

sx

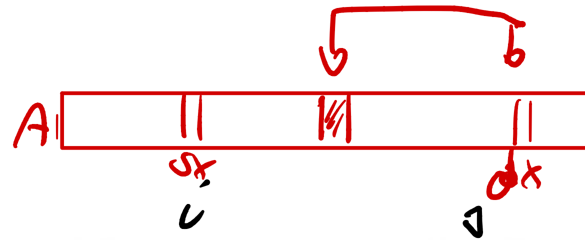
$[4, \boxed{3}, 2]$

distribution rx-1

$[2\ \boxed{3}\ 4]$

stop stop

```
QuickSort( a, sinistra, destra ):
     ⟨pre: 0 ≤ sinistra, destra ≤ n − 1⟩
  IF (sinistra < destra) {
    scegli pivot nell'intervallo [sinistra...destra];
    rango = Distribuzione( a, sinistra, pivot, destra );
    QuickSort( a, sinistra, rango-1 );
    QuickSort( a, rango+1, destra );
  }
```



scambia (i, j):
temp = A[i]
A[i] = A[j]
A[j] = temp

```
Distribuzione( a, sx, px, dx ):        ⟨pre: 0 ≤ sx ≤ px ≤ dx ≤ n − 1⟩
  IF (px != dx) Scambia( px, dx );
  i = sx;
  j = dx-1;
  WHILE (i <= j) {
    WHILE ((i <= j) && (A[i] <= A[dx]))
      i = i+1;
    WHILE ((i <= j) && (A[j] >= A[dx]))
      j = j-1;
    IF (i < j) Scambia( i, j );
  }
  IF (i != dx) Scambia( i, dx );
  RETURN i;
```

pivot

pivot

gdb debug

$i > j$

oppure

$A[i] > A[dx]$ e
$A[j] < A[dx]$

$$\begin{array}{ccccccc} & & sx & 1 & & & dx \\ A & = & [6, & \boxed{5}, & 3, & 7, & 2] \end{array}$$

$$\begin{array}{cccccc} A & [6, & 2, & 3, & 7, & \boxed{5}] \\ & i & & j & & \end{array}$$

$$\begin{array}{cccccc} A & [3, & 2, & 6, & 7, & \boxed{5}] \\ & & j & & & dx \\ & & & i & & \end{array} \rightsquigarrow [3, 2, \boxed{5}, 7, 6]$$

$n = dx - sx + 1$ elementi da ordinare

Distribuzione(a, sx, px, dx) richiede $O(n)$ tempo

```
Distribuzione( a, sx, px, dx ):                  ⟨pre: 0 ≤ sx ≤ px ≤ dx ≤ n - 1⟩
O(1)  IF (px != dx) Scambia( px, dx );
O(1)  i = sx;
O(1)  j = dx-1;
      WHILE (i <= j) {
                                 O(1)
        WHILE ((i <= j) && (A[i] <= A[dx]))
O(1)      i = i+1;
        WHILE ((i <= j) && (A[j] >= A[dx]))
O(1)      j = j-1;
O(1)    IF (i < j) Scambia( i, j );
      }
O(1)  IF (i != dx) Scambia( i, dx );
      RETURN i;
```

Il costo dei cicli while domina

1 while {
2 while ;
3 while ;
}

t_1 = # iterazioni 1
t_2 = " 2
t_3 = " 3

$n = dx - sx + 1$

$$t_1 \le n + t_2 + t_3 \le 3n \qquad (t_2 + t_3 \le n)$$

i++ al più n volte
j++ al più n volte

Possiamo concludere che globalmente ci sono $O(n)$ iterazioni (alcune annidate) che cumulativamente costano $O(n)$ tempo.

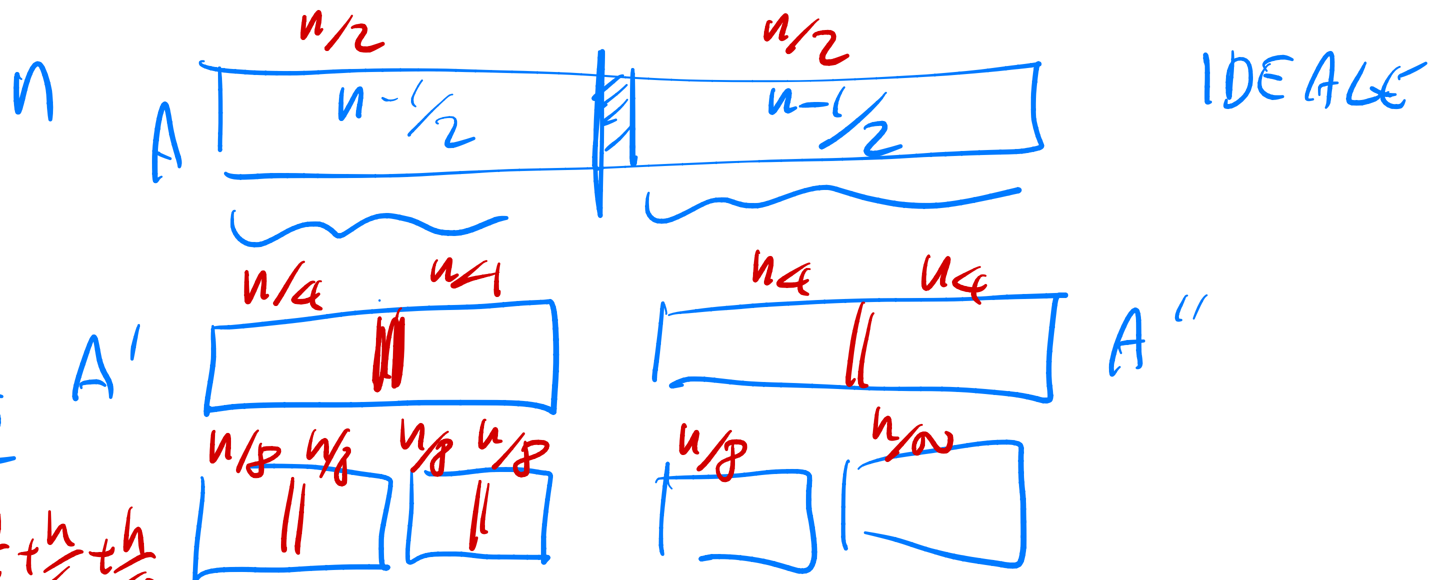
```
QuickSort( a, sinistra, destra ):
      ⟨pre: 0 ≤ sinistra, destra ≤ n - 1⟩
  IF (sinistra < destra) {
O(1) scegli pivot nell'intervallo [sinistra...destra];
O(n) rango = Distribuzione( a, sinistra, pivot, destra );
    QuickSort( a, sinistra, rango-1 );   ?
    QuickSort( a, rango+1, destra );
  }
```

CASO PESSIMO?

- se il pivot è sempre il minimo/massimo in $A[sx..dx]$
  QS usa $O(n)$ tempo per trovarsi poi
  con $A[sx..dx-1]$ oppure $A[sx+1..dx]$ da ordinare

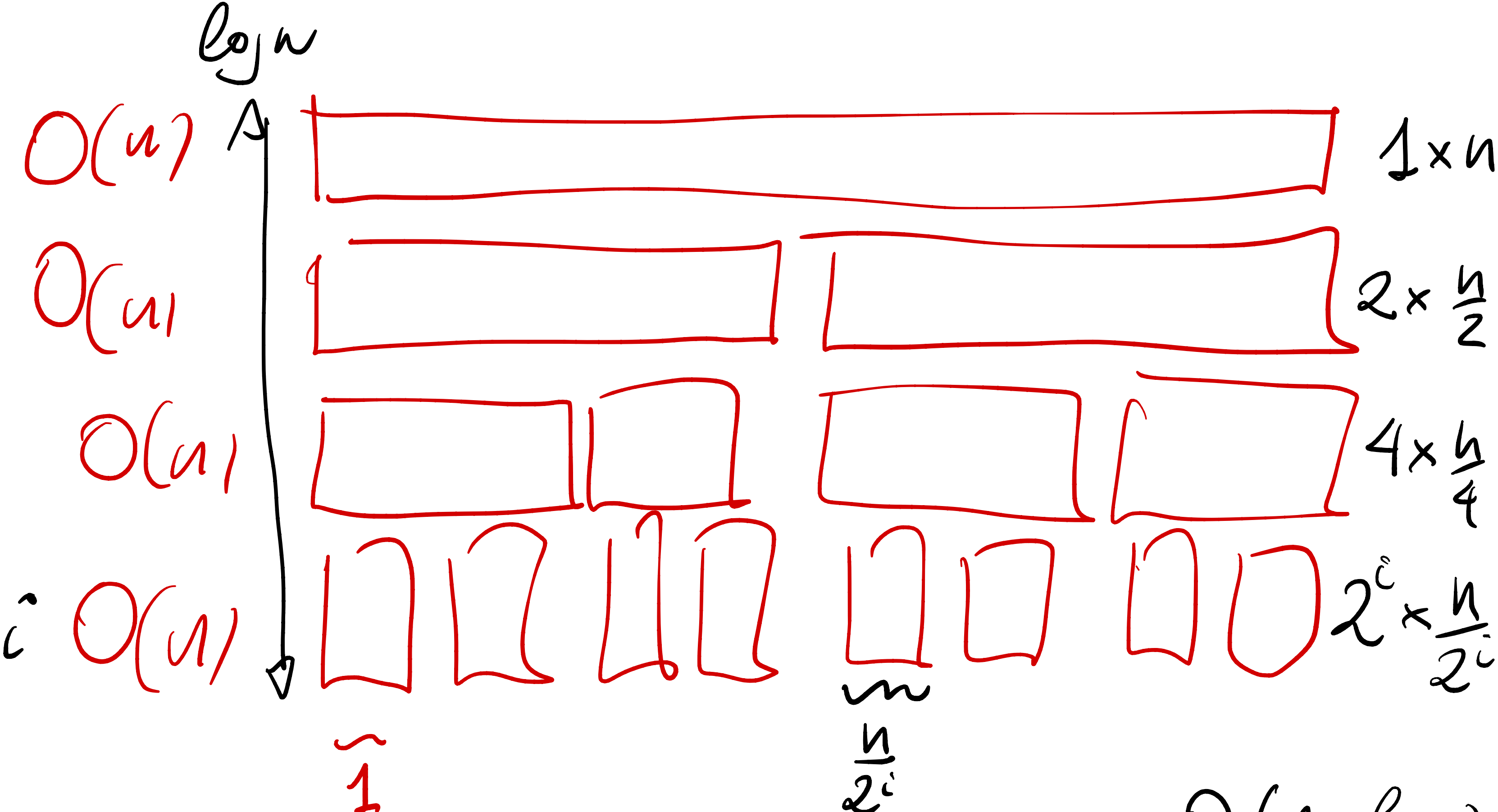
$$A = [\underbrace{3, 4}, \underset{\text{pivot}}{\boxed{2}}, \underbrace{7, 5, 22}]$$

CASO MEDIO? Il pivot va proprio modo a finire a metà strada tra sx e dx



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

$\frac{n}{4}+\frac{n}{4}+\frac{n}{4}+\frac{n}{4}$



$O(n \times \log n)$

STOP: $\frac{n}{2^i} = 1 \Rightarrow i = \log_2 n$

$O(n)$ $1 \times n$

$O(n-1)$ $1 \times (n-1)$

$O(n-2)$ $1 \times (n-2)$

$\vdots$

$$O\left(\sum_{i=1}^{n-1} (n-i)\right) = O(n^2)$$

$$n - i = 1 \Rightarrow i = n - 1$$