

Quick sort

Quicksort is a **divide-and-conquer algorithm**. It works by selecting a '**pivot**' element from the array and partitioning the other elements into two sub-arrays, according to whether they are less than or greater than the pivot.

Pseudocodice

```
procedura partizione (array, lo, hi): begin
    pivot = array[lo] array di int
    i = lo + 1 i < index
    j = hi - 1 j > index
    mentre(true):
        mentre(array[i] < pivot):
            i = i + 1
        questi sono 'do...while(...);'
        mentre(array[j] > pivot):
            j = j - 1
            lo è necessario incrementare i e
            decrementare j almeno di volta
        se ( i >= j):
            ritorna j
        Scambio (array[i], array[j])
    end
```

```
procedura quicksort (array, lo, hi): begin
    se ( lo >= 0 AND hi >= 0 AND lo < hi):
        p = partizione (array, lo, hi) → elemento che divide l'array in 2 sottovetori
        quicksort (array, lo, p)
        quicksort (array, p+1, hi)
    ritorna
end
```

Sorgente

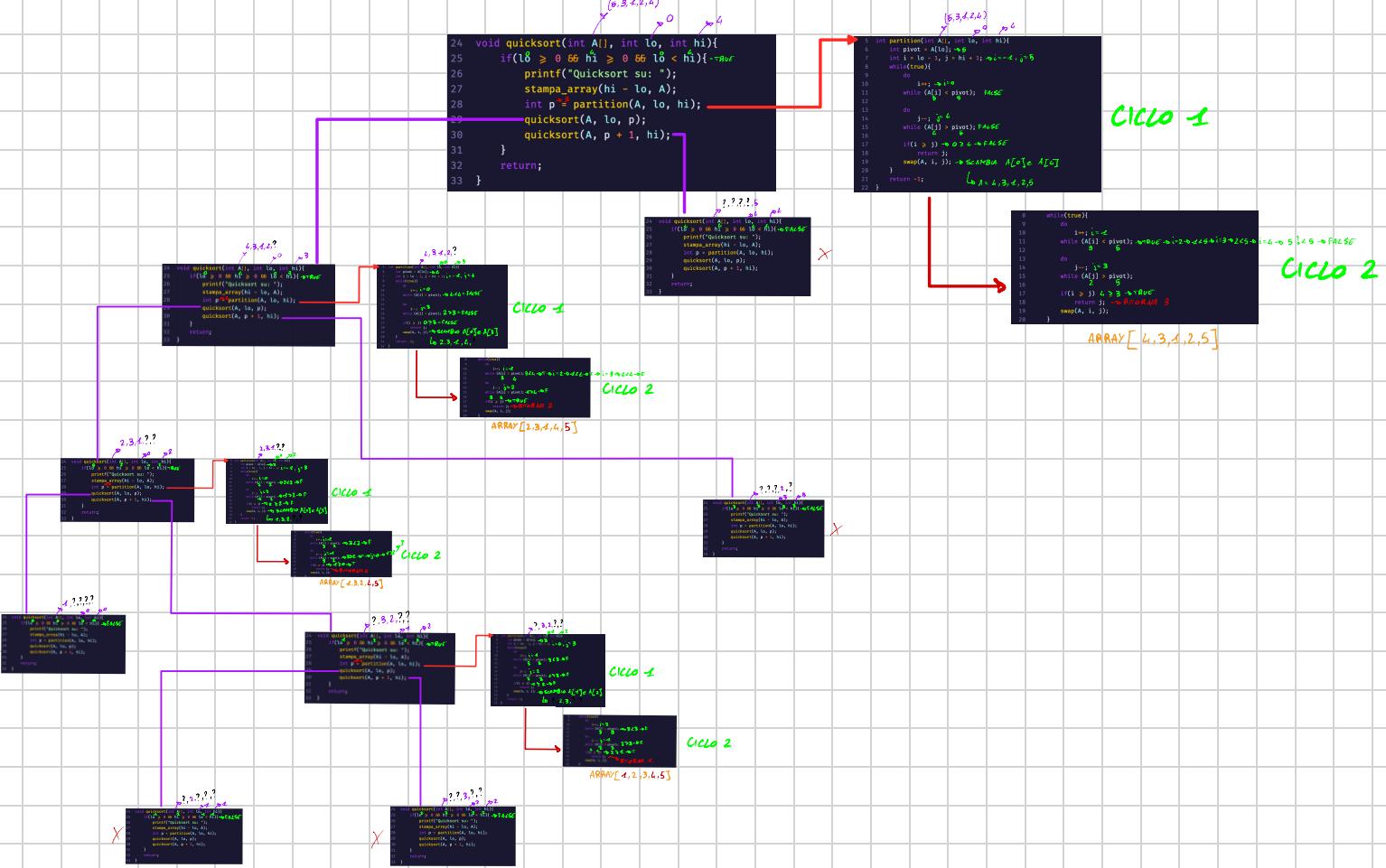
```
● ● ●
1 #include<stdio.h>
2 #include<basic/array.h>
3 #include<stdbool.h>
4
5 int partition(int A[], int lo, int hi){
6     int pivot = A[lo];
7     int i = lo - 1, j = hi + 1;
8     while(true){
9         do
10             i++;
11             while (A[i] < pivot);
12
13         do
14             j--;
15             while (A[j] > pivot);
16
17         if(i >= j)
18             return j;
19         swap(A, i, j); → invia i 2 elementi dell'array dati i loro indici
20     }
21     return -1;
22 }
23
24 void quicksort(int A[], int lo, int hi){
25     if(lo >= 0 && hi >= 0 && lo < hi){
26         printf("Quicksort su: ");
27         stampa_array(hi - lo, A);
28         int p = partition(A, lo, hi);
29         quicksort(A, lo, p);
30         quicksort(A, p + 1, hi);
31     }
32     return;
33 }
34
35 int main (void) {
36     int size = 0;
37     printf("Inserire la dimensione dell'array: ");
38     scanf("%d", &size);
39     int A[size];
40     printf("Inserire gli elementi:\n");
41     leggi_array(A, size);
42     printf("\nArray non ordinato: ");
43     stampa_array(size, A);
44
45     quicksort(A, 0, size - 1);
46
47     printf("Array ordinato: ");
48     stampa_array(size, A);
49     return 0;
50 }
```

Output

```
● leonardo@MacBook-Air-di-Leonardo ~ % gcc -o quick_sort.exe quick_sort.c
● leonardo@MacBook-Air-di-Leonardo ~ % ./quick_sort.exe
Inserire la dimensione dell'array: 5
Inserire gli elementi:
2
3
1
5
4
Array non ordinato: 2 3 1 5 4
Quicksort su: 2 3 1 5
Quicksort su: 1 3 2
Quicksort su: 1 2
Quicksort su: 1
Array ordinato: 1 2 3 4 5
```

Esempio:

$$A = \begin{pmatrix} 5 & 3 & 1 & 2 & 4 \end{pmatrix} \rightarrow \text{size} = 5$$



Conclusion:

Per ogni chiamata a quicksort che rispetti la condizione lo <hi la funzione 'partition' esegue almeno uno scambio. In ~~avanguardia~~ ho segnato lo stato dell'array al termine di ogni partizione. Gli elementi in ~~rosso~~ sono quelli non interessati da tale chiamata (e di cui non sappiamo per certo il valore visto che non conosciamo l'avanzamento delle altre chiamate a 'quicksort', motivo per il quale tra gli argomenti ho sostituito tali valori con '?')