Che funzioni vanno implementate?

`quicksort`

- Questa funzione è quella che viene chiamata dal programma.
- Chiama `partition()` usando i bounds che vengono passati alla funzione, e si salva il pivot che partition restituisce;
- Chiama poi se stessa ricorsivamente due volte:
  - Una sull'insieme dei valori tra lower bound e pivot;
  - Una sull'insieme dei valori tra pivot e upper bound;

`partition`

- Questa funzione viene chiamata esclusivamente da `quicksort()`
- Si occupa di dividere una sezione dell'array in due parti, una con gli elementi più grandi del pivot, una con gli elementi più piccoli;
- Modifica l'array in place e ritorna la posizione finale del pivot.

quicksort()

```
quicksort:
    bltu a3, a2, quicksort_exit      # if (lo >= hi) we just return

    # save stuff in the stack
    addi sp, sp, -32
    sd ra, 0(sp)
    sd s10, 8(sp)       # s10 is going to hold lo
    sd s11, 16(sp)      # s11 is going to hold hi
    sd s9, 24(sp)       # s9 is going to hold the pivot

    # hold lo and hi
    mv s10, a2          # s10 <- lo
    mv s11, a3          # s11 <- hi

    # call partition
    jal ra, partition

    # save the pivot on s9
    mv s9, a0

    # s9 = pivot
    # s10 = lo
    # s11 = hi
    # recursively call quicksort on both subarrays
    addi a3, s9, -1         # hi = pivot (-1)
    mv a2, s10              # lo = lo
    jal ra, quicksort       # quicksort(a1, lo, pivot-1)

    addi a2, s9, 1          # lo = pivot (+1)
    mv a3, s11              # hi = hi
    jal ra, quicksort       # quicksort(a1, pivot+1, hi)

    # load stuff back from the stack
    ld ra, 0(sp)
    ld s10, 8(sp)
    ld s11, 16(sp)
    ld s9, 24(sp)
    addi sp, sp, 32
quicksort_exit:
    ret
```

```
def quickSort(arr,low,high):
    if low < high:
        # pi is partitioning index, arr[p] is now
        # at right place
        pi = partition(arr,low,high)

        # Separately sort elements before
        # partition and after partition
        quickSort(arr, low, pi-1)
        quickSort(arr, pi+1, high)
```