Quicksort

Lab 6

Quick Sort

- One of the fastest Sorting algorithm - based on Divide and conquer approach

- 3 steps:
  - Choose a value to be sorted called Pivot
  - Partition such that elements to the left of the pivot are smaller and right are greater than the pivot
  - Recursively call quicksort on left and right subarrays

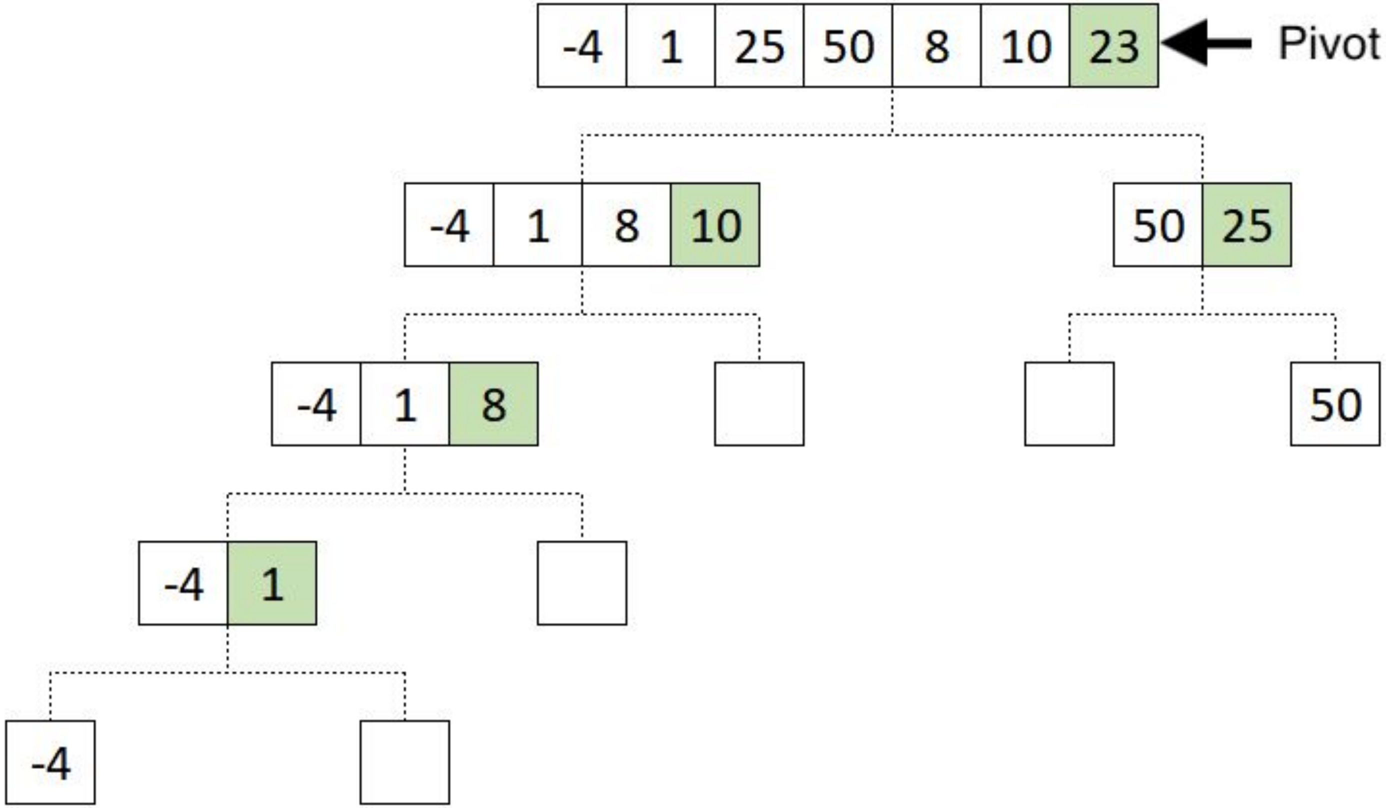
- Choice of pivot affects the efficiency and not correctness of the algorithm
  - Worst case
  - Average case
  - Why we randomize the pivot?

Choosing a pivot

- The algorithm will work correctly no matter which element you choose as the pivot.

- Choosing first or last element as pivot.
  - How does this affect the efficiency when the array is sorted?
  - Pivot consistently divides the array into highly imbalanced subarrays.

- A balanced pivot helps achieve optimal time complexity.
  - It is better if the pivot divides up the array into roughly equal partitions.


-4
1
25
50
8
10
23
Pivot
-4
1
8
10
50
25
-4
1
8
50
-4
1
-4

Partition() pseudocode

```
partition(low, high, array){
    j = low
    pivotIndex = high
    for (i = low ; i < high; i ++){
        if array[i] < pivot{
            swap(array[i], array[j])
            j ++;
        }
    }
    swap(array[j], array[pivotIndex])
    return j
}
```

Quicksort() calling recursively

```
quicksort(){
        if low < high:
                partitionindex = partition(low, high, arr)
                quicksort(low, partitionindex - 1, arr)
                quicksort(partitionindex + 1, high, arr)
}
```

Implementation

- Download project6.tar from camino
- Change getElements to return a sorted array by implementing the quicksort algorithm
  - Suggestions:
    - Create the private partition function and recursive quicksort function that implements the quicksort algorithm
      - int partition()
      - void quicksort()
    - Call the quicksort function before returning the array in getElements

- Notes: do not call qsort() function from c library!

Note:

- Specify the time complexity for getElements() after you include the quicksort
- Make 2 separate functions partition() and quicksort() and mention their time complexities

Test

- Use “-l” with unique program to test getElements:

```
./unique -l /scratch/coen12/GreenEggsAndHam.txt
```

- You may test with other text files but the output should be the unique words in order.