QuickSelect

Warm-up

How can I find the 3rd smallest element in the next array?

data = [10, 3, 6, 9, 2, 4, 15, 23], K = 3rd

Output: 4

The partition method could help.

Divide and Conquer

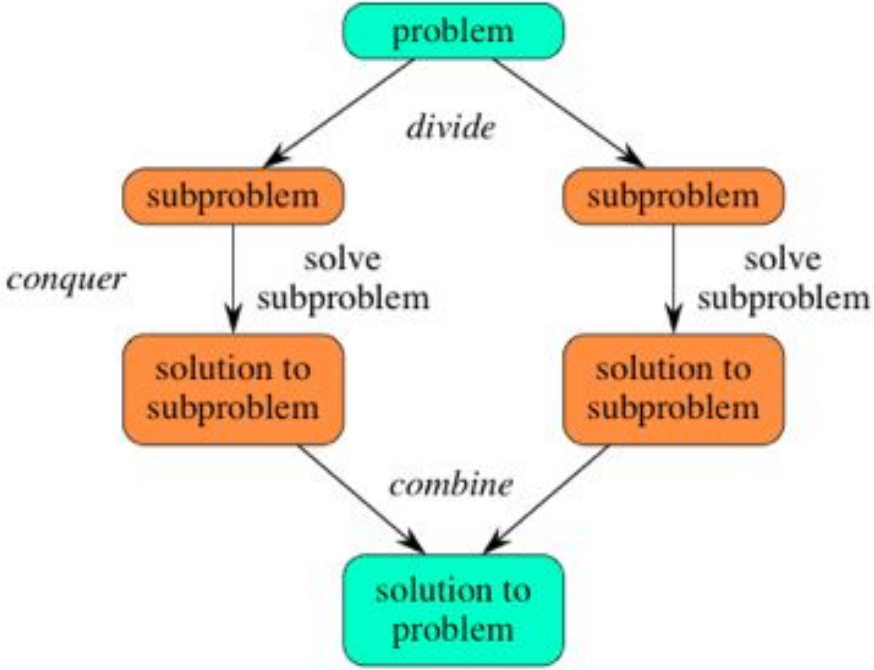
Some algorithms, such as Merge sort and QuickSort and QuickSelect employ a common algorithmic paradigm based on recursion.

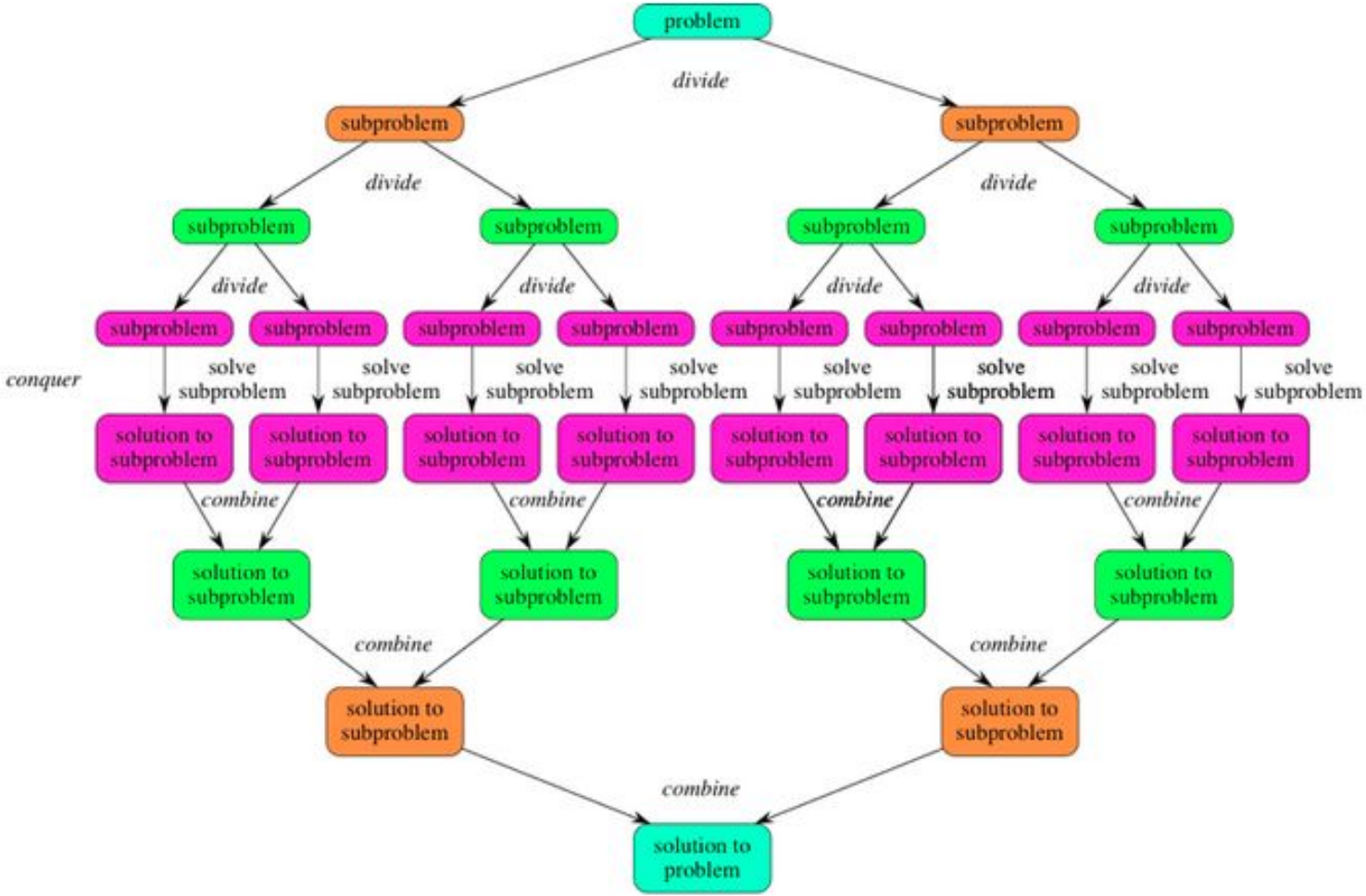
A divide-and-conquer algorithm recursively breaks down a problem into two or more sub-problems of the same or related type, until these become simple enough to be solved directly. (Wikipedia)

Divide and conquer algorithm parts

1. **Divide** the problem into a number of subproblems that are smaller instances of the same problem.
2. **Conquer** the subproblems by solving them recursively.
3. **Combine** the solutions to the subproblems into the solution for the original problem.

Divide and Conquer




problem
divide
subproblem
subproblem
divide
divide
subproblem
subproblem
subproblem
subproblem
divide
divide
divide
divide
subproblem
subproblem
subproblem
subproblem
subproblem
subproblem
subproblem
subproblem
conquer
solve subproblem
solve subproblem
solve subproblem
solve subproblem
solve subproblem
solve subproblem
solve subproblem
solve subproblem
solution to subproblem
solution to subproblem
solution to subproblem
solution to subproblem
solution to subproblem
solution to subproblem
solution to subproblem
solution to subproblem
combine
combine
combine
combine
solution to subproblem
solution to subproblem
solution to subproblem
solution to subproblem
combine
combine
solution to subproblem
solution to subproblem
combine
solution to problem

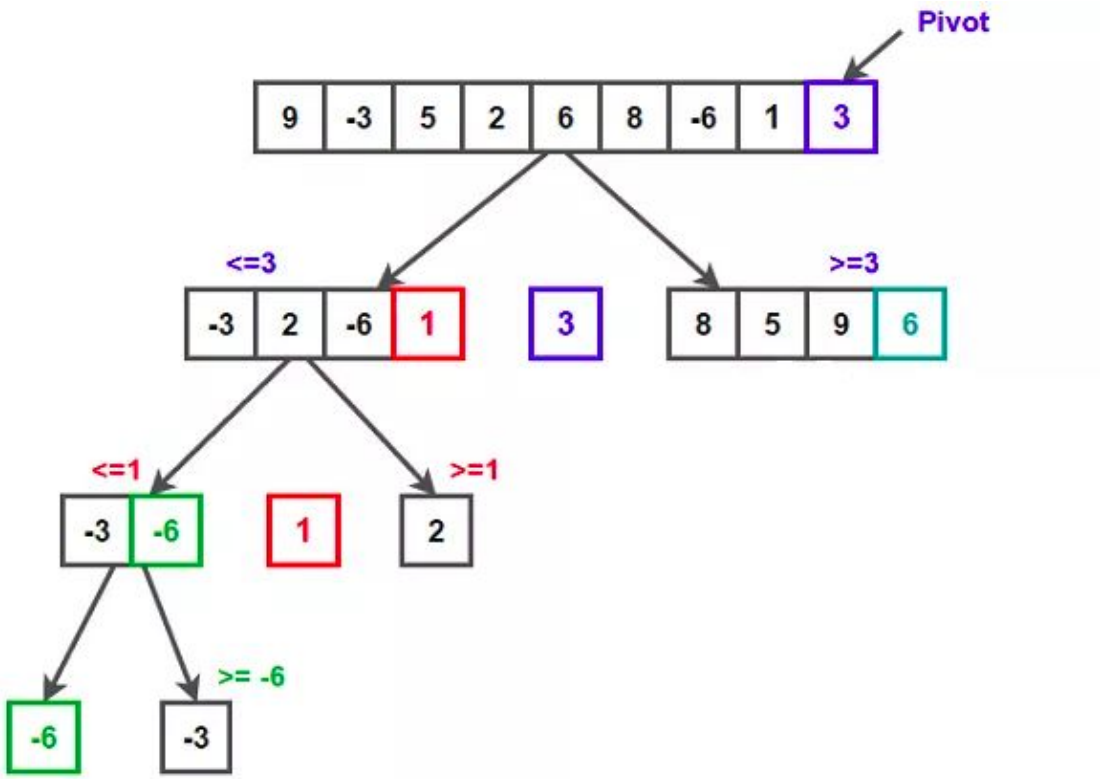
Quick Select

It is a selection algorithm. It looks for the k-th smallest element in an unsorted array.

It uses a partition strategy to find if index of the partitioned element is more than k, then we recur for the left part. If index is the same as k, we have found the k-th smallest element and we return. If index is less than k, then we recur for the right part.

Using a random pivot when defining partitions we can avoid the worst case (largest/smallest element picked as pivot).

Quick Select



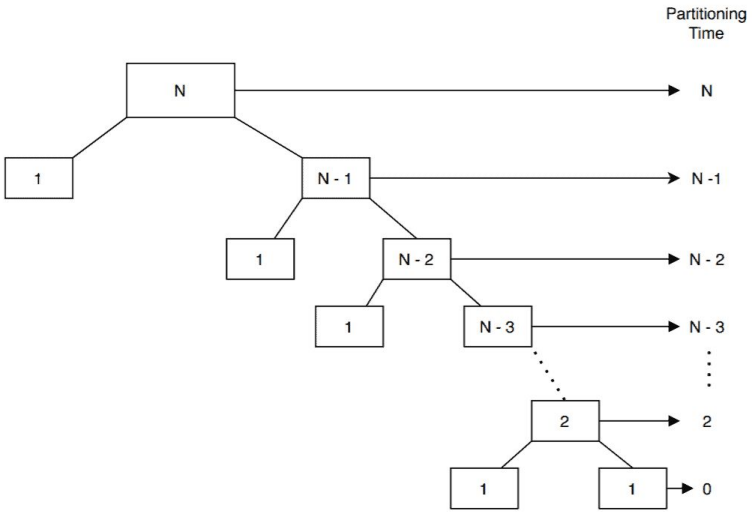
Quick Select - Algorithm

QuickSelect method to find the Kth smallest element

```
pivot  = partition(data, start, end)
if pivot > k -1
    Recursive call to your QuickSelect - parameters: start=?, end=?
Else if pivot < k -1
    Recursive call to your QuickSelect - parameters: start=?, end=?
Else
    Kth element found return  the value [k-1]
```

Quick Select - Time Complexity

Worst case: larger/smallest element picked as pivot



O(N^2)

The height of the tree will be n and in top node we will be doing N operations

then n-1 and so on until 1

Quick Select - Time Complexity

Best Case: when we partition the list into two halves and continue with only the half we are interested in.

N + N/2 + N/4 + N/8 + N/16

The sum of that series will never reach 2*N.

O(n)

Coding Time!

Write your code here: classwork/xx_quick/Quick.java

You must implement the following method:

```
public static int quickSelect(int[] data, int k, int start, int end){
}
```

Copy the partition method to your Quick.java

Call the method like this:

```
// find the third smallest element (test for different Kth elements)
quickSelect(yourArray, 3, 0, yourArray.length-1));
```