Amena Tajammul
CS430 Due: 02/20/2024

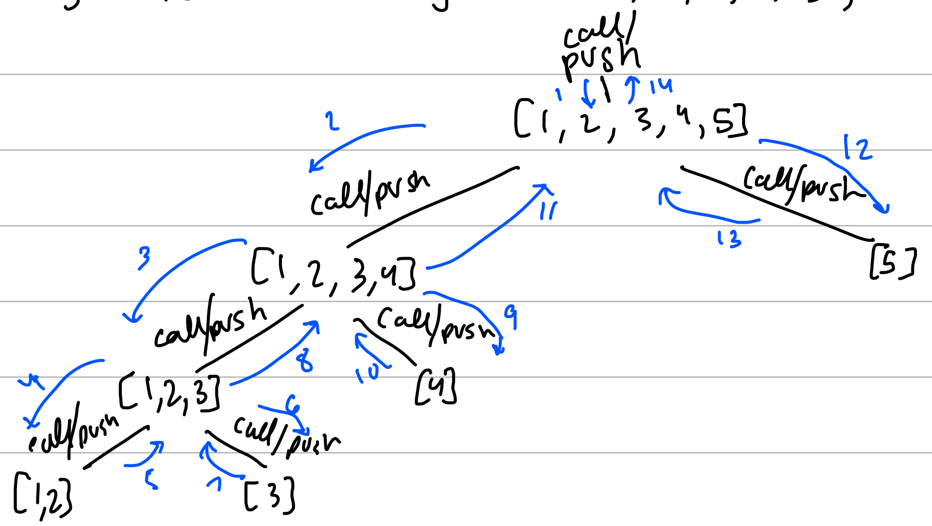
CS 430 HW#3

① a. We know that QUICKSORT correctly sorts the array. The only difference between QUICKSORT and QUICKSORT' is that QUICKSORT' doesn't do a second recursive call. However, the second run of the while loop calls the resursion with the same values that the 2nd run would have.

2nd rec. = QUICKSORT(A, q+1, r)

2nd while run = QUICKSORT'(A, p, q−1), where p=q+1 and r=q-1

b. Say we have an array A=[1,2,3,4,5], n=5



The stack depth of QUICKSORT' is $\Theta(n)$ because the depth is reached at the fourth recursive call, which is 5−1=4 aka n−1.

c. To make the worst case stack depth $\Theta(\log n)$, we could add a comparison between r-(q+1) and q−1−P to see which is smaller. Then, we can use the Partition on the smaller array as it would make the depth $\Theta(\log n)$. The expected run time of the algorithm is $O(n\log n)$.

② The smallest possible depth of a leaf node in a decision tree for a comparison sort is n−1. In the best case, the array is already in order, but we still need to make n−1 comparisons to see this. In the worst case, the array is in reverse order, so we need to compare each element with every other element. Thus, the largest possible depth of a leaf node in a decision tree for a comparison sort is n^2.

③ Counting sort is not the best choice to sort the array. This is because counting sort only works well when the elements of the array are small positive integers. The range of the provided array is from -29864 to 89926187. Although the number of elements is small, the large range makes it hard for counting sort to work with.

④
```
KthLargestElement (A, K){
    if K=0
        return 0
    else if  K > length[A]
        return 0
    QUICKSORT(A, 0, length[A])
    m= length[A]-K
    n = A[m]
    return n
}
```