**Quick sort:**

Functions by taking an array, splitting it up into three smaller arrays:

Left, right, and middle, with the middle one consisting only of one entry. The function enters recurrence and does the same for left and right. At the lowest level of recurrence, the arrays get sorted, and sent up to the upper layers, repeating the sorting process on the sorted subarrays.

**Partition function (How shit gets sorted):**

It should also be noted that the function uses a pivot-selection function to determine q. It’s really fucky tbh. The jist of it is, you start by setting a var=i=p-1, and a var=x=A[r]. You then use a for loop, where j=p..r-1. For each j, you check if A[j] <= x. If it is, increment i by one and swap A[i] with A[j]. Otherwise, swap A[i+1] with A[r].

**Runtimes:**

Based on whether or not there is a balanced or unbalanced amount of entries in the array being sorted on, the quick sort algorithm will have vastly different runtimes. In this case, a truly balanced amount refers to cases where subproblems where one subsubproblem has exactly half the size of the subproblem, while the other subsubproblem has exactly one less than half the size.

Long story short, worst case scenario runtime is while best case is