

PAY due tomorrow
PAY Late/Reschedule → due tomorrow

Quick sort → in-place

```

Sorting Quickly
public class SortQuickly {
    public static void swap(String[] array, int i1, int i2) {
        String temp = array[i1];
        array[i1] = array[i2];
        array[i2] = temp;
    }

    public static int partition(String[] array, int low, int high) {
        int pivotStartIndex = high - 1;
        String pivot = array[pivotStartIndex];
        int smallerBefore = low, largerAfter = high - 2;
        while (smallerBefore <= largerAfter) {
            if (array[smallerBefore].compareTo(pivot) < 0) {
                smallerBefore++;
            } else {
                swap(array, smallerBefore, largerAfter);
                largerAfter--;
            }
        }
        swap(array, smallerBefore, pivotStartIndex);
        return smallerBefore;
    }

    public static void qsort(String[] array, int low, int high) {
        if (high - low <= 1) { return; }
        int splitAt = partition(array, low, high);
        qsort(array, low, splitAt);
        qsort(array, splitAt + 1, high);
    }

    public static void sortD(String[] array) {
        qsort(array, 0, array.length);
    }

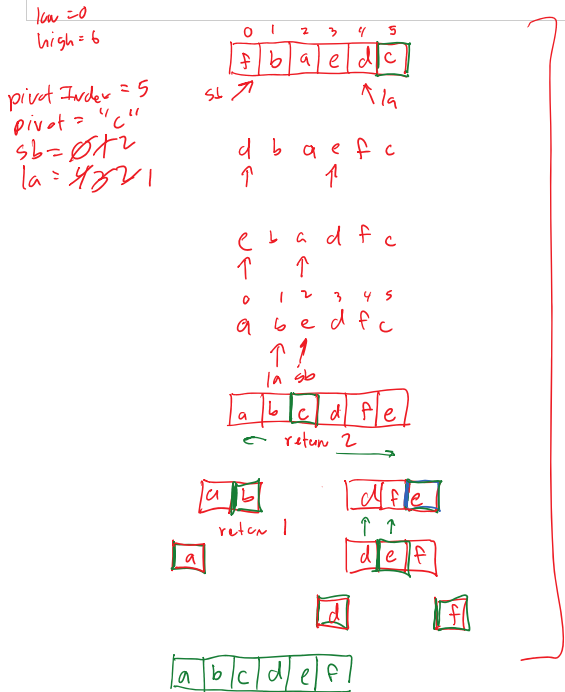
    public static void main(String[] args) {
        String[] str = {"f", "b", "a", "e", "d", "c"};
        int[] result = SortQuickly.sortD(str);
        System.out.println(Arrays.deepToString(result));
    }
}
    
```

Draw the picture of sortD()

What is the tight bound of sortD:

Best case: $\Theta(N \log_2(N)) \rightarrow$ median partition value at every level

Worst case: $\Theta(N^2) \rightarrow$ sorted array

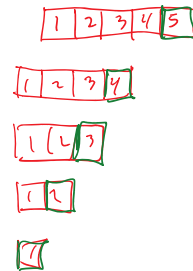


$N = 6$
height = 3
 $\log_2(N)$

$\Theta(N \log_2(N))$

1
2
3

Worst case → sorted array



$N = 5$
levels = 5
 $\Theta(N^2)$

median value as pivot

