Noah Buchanan

Problem Set 3
Algorithms

October 3, 2020

UAWord and Quicksort

```
Partition(UAWord[] A, int p,r)

        q = Median(p,r,(p+r)/2)
        swap(q,r)
        i = p - 1
        for (j to r - 1)
                if(A[j].getCount() > A[r].getCount())
                        i = i + 1
                        swap(i,j)
        swap(i+1,r)
        return i + 1

swap(int x,y)
        temp = A[x].getWord()
        A[x].setWord() = A[y].getWord()
        A[y].setWord() = temp

Median(int a,b,c)
        if((A[a] > A[b] and A[a] < A[c]) or (A[a] < A[b] and A[a] > A[c]))
                return a
        else if((A[b] > A[c] and A[b] < A[a]) or (A[b] < A[c] and A[b] > A[a]))
                return b
        else if((A[c] > A[b] and A[c] < A[a]) or (A[c] < A[b] and A[c] > A[a]))
                return c
```

1. Assuming we picked 5 as the partition the problem that would arise from this is that 5 is no pivot at all, it is the largest value, the same could happen

for 1 as well. This creates a completely unbalanced set of subarrays, one containing absolutely no values, and the other containing n-1. The recurrence for the running time in a worst case where it repeatedly picks a bad partition like this is as follows:

$$\begin{aligned} T(n) &= T(n-1) + T(0) + \Theta(n) \\ &= T(n-1) + \Theta(n) \\ T(n-1) &= T((n-1)-1) + \Theta(n) + \Theta(n) \\ &= T((n-1)-2) + \Theta(n) + \Theta(n) + \Theta(n) \\ &= T(n-n) + \Theta(n) * n \\ &= n^2 \end{aligned}$$

As you can see, at n iterations, the algortihm has compared values n^2 times, hence the worst case runtime of $O(n^2)$

2. A terrible split, (0 and n) values in both subarrays, followed by a good split (n/2 and n/2) produces a running time much closer to $n \lg n$. And even on average a 9 to 1 split still runs much closer to $n \lg n$ than n^2. Suppose we get the worst case scenario of a 9 to 1 ratio split on every split. The reccurence to represent this is as follows:

$$\begin{aligned} T(n) &= T(9n/10) + T(n/10) + \Theta(1) \\ T(n-1) &= T(9(n-1)/10) + T((n-1)/10) + \Theta(1) + \Theta(1) \\ &= T(9(n-1) - 1/10) + T((n-1) - 1/10) + \Theta(1) + \Theta(1) + \Theta(1) \\ &= T(9(n-n)/10) + T((n-n)/10) + \Theta(1) * n \\ &= T(9(0)/10) + T(0/10) + \Theta(1) * n \\ &= \Theta(1) * n \\ &= n \end{aligned}$$

Even at the second to worst case of a repeated 9 - 1 split, it runs far closer to lgn n time than it does n^2 at $\Theta(n)$. At the same time, this is not the Average case however, the Average case is far closer to an even split than a 9-1 is, so even with the occasional 9-1 split, with the rest of the splits being on average closer to the expected values then it will lead to a performance of $\Theta(n \lg n)$

Algorithm Design

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
```

```
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class UASort {

        /*  Student Name:        Noah Buchanan
         *  Username:               ua505               <--- this needs to be correct
         *  Date:                       September 16
         *  Class:          CS 3103 - Algorithm Design
         *  Filename:       UASort.java
         */

        String[] A;

        public static void main(String[] args) throws IOException {

                if (args.length < 4) {
                        System.out.println(
                                        "Incorrect arguements, try: java UASort (input d

                } else {

                        if (args[2].equals("numeric") && args[3].equals("descending")) {

                                UASort s = new UASort();
                                File[] x = new File(args[0]).listFiles();

                                for (int i = 0; i < x.length; i++) {

                                        s.loadData(x[i].getAbsolutePath());
                                        s.QuickSortNumerics(s.A, 0, s.A.length - 1);
                                        s.Flip(s.A);
                                        File y = new File("/home/ua505/ps4/" + args[1] +
                                        y.createNewFile();
                                        s.writeData("/home/ua505/ps4/" + args[1] + "/" +
                                }
                        } else if (args[2].equals("numeric") && args[3].equals("ascendin

                                UASort s = new UASort();
                                File[] x = new File(args[0]).listFiles();

                                for (int i = 0; i < x.length; i++) {

                                        s.loadData(x[i].getAbsolutePath());
```

```
                        s.QuickSortNumerics(s.A, 0, s.A.length - 1);
                        File y = new File("/home/ua505/ps4/" + args[1] +
                        y.createNewFile();
                        s.writeData("/home/ua505/ps4/" + args[1] + "/" +
                }
        } else if (args[2].equals("text") && args[3].equals("descending"

                UASort s = new UASort();
                File[] x = new File(args[0]).listFiles();

                for (int i = 0; i < x.length; i++) {

                        s.loadData(x[i].getAbsolutePath());
                        s.QuickSortText(s.A, 0, s.A.length - 1);
                        s.Flip(s.A);
                        File y = new File("/home/ua505/ps4/" + args[1] +
                        y.createNewFile();
                        s.writeData("/home/ua505/ps4/" + args[1] + "/" +
                }
        } else if (args[2].equals("text") && args[3].equals("ascending"))

                UASort s = new UASort();
                File[] x = new File(args[0]).listFiles();

                for (int i = 0; i < x.length; i++) {

                        s.loadData(x[i].getAbsolutePath());
                        s.QuickSortText(s.A, 0, s.A.length - 1);
                        File y = new File("/home/ua505/ps4/" + args[1] +
                        y.createNewFile();
                        s.writeData("/home/ua505/ps4/" + args[1] + "/" +
                }
        } else {
                System.out.println("Arguements not recognized");
        }

    }
}

public void writeData(String output) throws IOException {

    BufferedWriter bw = new BufferedWriter(new FileWriter(output));

    for (int i = 0; i < A.length; i++) {
            bw.write(A[i] + " ");
    }
```

```
        bw.close();
}

public void loadData(String input) throws IOException {

        BufferedReader br = new BufferedReader(new FileReader(input));

        String line = br.readLine();

        A = line.split(" ");

        br.close();
}

public void QuickSortText(String[] A, int p, int r) {
        if (p < r) {
                int q = PartitionText(A, p, r);
                QuickSortText(A, p, q - 1);
                QuickSortText(A, q + 1, r);
        }
}

public int PartitionText(String[] A, int p, int r) {
        String x = A[selectPivotText(A, p, r)];
        Swap(r, selectPivotText(A, p, r));
        int i = p - 1;
        for (int j = p; j < r; j++) {
                if (A[j].compareTo(x) <= 0) {
                        i = i + 1;
                        Swap(i, j);
                }
        }
        Swap(i + 1, r);
        return i + 1;
}

public void QuickSortNumerics(String[] A, int p, int r) {
        if (p < r) {
                int q = PartitionNumerics(A, p, r);
                QuickSortNumerics(A, p, q - 1);
                QuickSortNumerics(A, q + 1, r);
        }
}

public int PartitionNumerics(String[] A, int p, int r) {
        int x = Integer.parseInt(A[selectPivotNumerics(A, p, r)]);
```

```
        Swap(r, selectPivotNumerics(A, p, r));
        int i = p - 1;
        for (int j = p; j < r; j++) {
                if (Integer.parseInt(A[j]) <= x) {
                        i = i + 1;
                        Swap(i, j);
                }
        }
        Swap(i + 1, r);
        return i + 1;
}

public int selectPivotNumerics(String[] A, int p, int r) {
        int q = (p + r) / 2;
        if (Integer.parseInt(A[p]) < Integer.parseInt(A[r]) && Integer.parseInt(A
                return r;
        } else if (Integer.parseInt(A[p]) < Integer.parseInt(A[q]) && Integer.par
                return q;
        } else {
                return p;
        }
}

public int selectPivotText(String[] A, int p, int r) {
        int q = (p + r) / 2;
        if (A[p].compareTo(A[r]) >= 0 && A[r].compareTo(A[q]) >= 0) {
                return r;
        } else if (A[p].compareTo(A[q]) >= 0 && A[q].compareTo(A[r]) >= 0) {
                return q;
        } else {
                return p;
        }
}

public void Swap(int x, int y) {
        String hold = A[x];
        A[x] = A[y];
        A[y] = hold;

}

public void Flip(String[] x) {
        String temp;
        for (int i = 0; i < x.length / 2; i++) {
                temp = x[i];
                x[i] = x[x.length - 1 - i];
```

```
                        x[x.length - 1 - i] = temp;
                }
        }

}
```