String Splitting Algorithm

Talia Hicks and Steven Rosendahl

We chose to implement the string breaking algorithm. To do this, we built a recursive algorithm that looks down the left and right side of the string, using the i^{th} breakpoint as a starting index. The program loops through all possible starting indexes, and takes the least of them.

```
for(int i = 0; i < cuts.length; i++){
    cost = Math.min(nval = splitStringRec(cuts, n, i), cost);
}
```

This loop runs n times, where n is the size of the list of breakpoints. The function *splitStringRec* is a recursive function that runs down the left and right sides of the string, trying out different breaks.

```
//Looping down the left
for(int j = 0; j < cpy.length; j++){
    cutlen = Math.min(splitStringRec(cpy, first, j), cutlen);
}

//Looping down the right
for(int i = 0; i < cpy.length; i++){
    cutr = Math.min(splitStringRec(cpy, newcost, i), cutr);
}
```

The loop that runs through the left breakpoints runs once the first time through, then twice on the second time through, and so on until it has run $n-1$ times. Similarly, the loop that runs through the right breakpoints runs $n-1$ times, then $n-2$, $n-3$, and so on until it only runs once. Since each loop takes the minimum of the cost, we have that only the minimum cuts will be preserved, while the rest are thrown away. The runtime of such an algorithm is close to

$$\begin{aligned}\sum_{n=0}^{i}\left(\sum_{k=0}^{n}(n-k)+\sum_{k=0}^{n}(k+n)\right) &= \sum_{n=0}^{i}\left(O(n^2)+O(n^2)\right)\\ &= \sum_{n=0}^{i}O(n^2)\\ &= O(n^3)\end{aligned}$$

To state the recurrence relation, let n be the length of the string, C be the array of cuts, and $i = |C|$. We want to build a definition such that we loop down the left half of the string, making cuts, which results in a new string of size $n-c$. Similarly, we want to move down the right side of the string, making cuts which result in strings of size $c-n$. We also want to "remove" the cuts that we have already made, so we create a new array that contains all the remaining cuts without the cut we just made (i.e. $C\backslash c_0$ where c_0 is the cut we just made).

$$\text{min\_cost}(n, C) = \begin{cases} n, & i = 1 \\ n + \min\begin{cases}\text{min\_cost}(n-c_0,\ C\backslash c_0)\\ \text{min\_cost}(c_0-n,\ C\backslash c_0)\end{cases}, & i > 1\end{cases}$$