Ally Smith

Homework 5

1. A certain string processing language allows the programmer to break a string into two pieces. It costs n units of time to break a string of n characters into two pieces, since this involves copying the old string. A programmer wants to break a string into many pieces, and the order in which the breaks are made can affect the total amount of time used. For example, suppose we wish to break a 20-character string after characters 3, 8, and 10. If the breaks are made in left-right order, then the first break costs 20 units of time, the second break costs 17 units of time, and the third break costs 12 units of time, for a total of 49 steps. If the breaks are made in right-left order, the first break costs 20 units of time, the second break costs 10 units of time, and the third break costs 8 units of time, for a total of only 38 steps.

   Give a dynamic programming algorithm that takes a list of character positions after which to break and determines the cheapest break cost in $O(n^3)$ time.

   Recursive Idea: Using the example from above, we can imagine a scenario in which we break up our string at the 10th index first for a cost of 20, leaving us with the remaining 10 characters to be split at indexes 8 and 3. We can continue doing this until we have no more splits left to make.

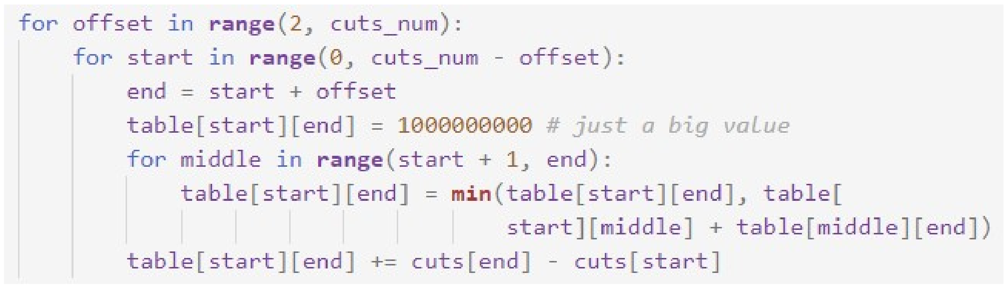
   Recursive Relation: The relation can be defined as follows, where $cost(i, j)$ is the cost to split a string starting at index i and going to j.
   $cost(i, j) = \min\{\text{length of substring} + cost(i, k) + cost(k, j), i < k < j\}$
   For all $j \leq i+1$, $cost(i, j) = 0$ since we cannot split the string at any value of k. This is our base case.

   Table: The values along the rows and columns depends on the split indexes, and the values that are blank depend on those indexes.

$\frac{i}{j}$	0	1	2	3	...	$n-1$	n
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2		0	0	0	0	0	0
3			0	0	0	0	0
...				0	0	0	0
$n-1$					0	0	0
n						0	0

Algorithm: The algorithm below depends on a value n, the length of the string, and a list of the indexes the user wishes to break the string on.

Once the algorithm has run, the minimum cost is in the bottom left corner of the table.



```python
for offset in range(2, cuts_num):
    for start in range(0, cuts_num - offset):
        end = start + offset
        table[start][end] = 1000000000 # just a big value
        for middle in range(start + 1, end):
            table[start][end] = min(table[start][end], table[
                                        start][middle] + table[middle][end])
        table[start][end] += cuts[end] - cuts[start]
```