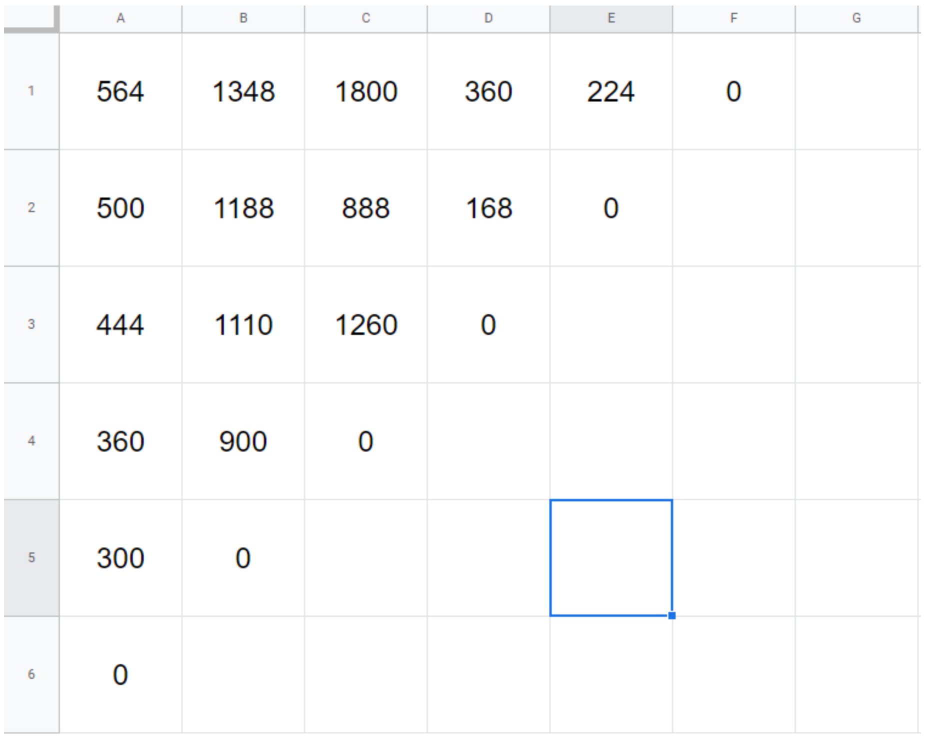
Noah Buchanan
Problem Set 6
Algorithms

October 22, 2020

I apologize if the source code is a bit hard to read in certain places, I couldn't reduce the tab size to be small enough in latex to still be acceptable and readable and have it all on the same line so some of the if statements and print statements are on two lines and it looks a little messy.

The minimum number of calculations possible with the optimal parenthesization is 564 calculations with a table shown below of how I got that answer.



	A	B	C	D	E	F	G
1	564	1348	1800	360	224	0	
2	500	1188	888	168	0		
3	444	1110	1260	0			
4	360	900	0				
5	300	0					
6	0						

UASpellCheck Class:

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.util.ArrayList;

public class UASpellCheck {

  String[] dictionary;

  public UASpellCheck(String filename) throws Exception {

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

    String line = "";
    String concat = "";

    while ((line = br.readLine()) != null) {

      concat += line + " ";
    }
    String[] dictionary = concat.split(" ");
    this.dictionary = dictionary;

  }



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

    UASpellCheck c = new UASpellCheck(args[0]);

    ArrayList<String> list = c.checkSpelling(args[1]);

    System.out.println("Word\t\tDistance");
    System.out.println("————————   —————————");

    for(int i = 0; i < list.size(); i++) {


      System.out.println(list.get(i) + "        \t" + c.editDistance(args[1],
          list.get(i), args[1].length(), list.get(i).length()));

    }
```

```
}



public boolean isValid(String word) {

  boolean valid = false;

  for (int i = 0; i < dictionary.length; i++) {
    if (word.equals(dictionary[i]))
      valid = true;

  }

  return valid;
}




public ArrayList<String> checkSpelling(String word) {

  ArrayList<String> list = new ArrayList<String>();

  int min = Integer.MAX_VALUE;
  for(int i = 0; i < dictionary.length; i++) {

    int val = editDistance(word,dictionary[i],word.length(),
                    dictionary[i].length());
    if(val < min) {

      list.clear();
      list.add(dictionary[i]);
      min = val;
    } else if(val == min) {

      list.add(dictionary[i]);
    }
  }
  if(word.length() >= 1) {
    return list;
  } else {
    return null;
  }

}




public int editDistance(String s1, String s2, int x, int y) {

  int table[][] = new int[x+1][y+1];

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

    for(int j = 0; j <= y; j++) {

      if(i==0) {

        table[i][j] = j;
      } else if(j == 0) {

        table[i][j] = i;
      } else if(s1.charAt(i - 1) == s2.charAt(j-1)) {

        table[i][j] = table[i-1][j-1];
      } else {

        if(minimum(table[i-1][j-1], table[i-1][j],
            table[i][j-1]) == table[i-1][j-1]) {

          table[i][j] = minimum(table[i-1][j-1],
              table[i-1][j], table[i][j-1]) + 2;

        } else {

          table[i][j] = minimum(table[i-1][j-1],
              table[i-1][j], table[i][j-1]) + 1;

        }

      }
    }
  }
  return table[x][y];
}
```

```
  public int minimum(int a, int b, int c) {

    if(a <= b && a <= c) {
      return a;
    } else if(b <= a && b <= c) {
      return b;
    } else {
      return c;
    }

  }

}
```