CS210 Final Assignment

Stephen Gallagher

Original Idea:

Randomly Generated Sentences Method

The original idea for the algorithm was to generate a pair of random sentences and compare the two. The highest score would be stored. The comparison was placed inside a while loop, which was terminated once a max score of a desired value was achieved.

```
while(maxScore < desiredmaxScore){
  //keep comparing until desiredmaxScore is achieved
}
```

I decided on having a sentence format:
Name (possessive) + noun + verb + adverb
Each part was sorted into sections, e.g. ‘names’ started at index 217 in the text file

e.g. “Alex’s toaster levitated gracefully.“

I wanted to get more comparisons done per while loop, so I wrote code to fill an array of length 100 with randomly generated sentences. The sentences in the array were then compared with every other sentence in the array, using an algorithm where element 0 is compared with element 1, 0 is compared with 2 etc.
This yielded faster and more consistent results but still struggled to get high scores. It could consistently get a score of around 19 rather quickly, but any score above that would take a lot of time.
I decided duplicate sentence pairs were affecting efficiency. The text file size was increased. The text file consists of 1,913 names, 188 nouns, 13 verbs and 12 adverbs, giving a total of 56,104,464 total sentence combinations.

Final Method - Iterative Sentences Method

To ensure no duplicate pairs were generated, I decided on employing an iterative method of generating sentences. I did this by designing a counter algorithm.

Sentence structure:

Name (possessive) + noun + verb + adverb

The counter would index through all the adverbs. Once it reached the limit of the adverbs, it would reset, and increment ‘verbs’ by one. Once ‘verbs’ reached its limit, ‘noun’ would increment once, etc.

```
/*counter algorithm, used to
  iterate through all possible
  sentences
*/
for(int i = 0; i < hashVal.length ; i++){
    String name = dic.getWord(names);
    String sentence = name.charAt(name.length() - 1) == 's' ? name + "' " : name + "'s " + dic.getWord(nouns) + " " + dic.getWord(verbs) + " " + dic.getWord(adverbs) + ".";
    hashVal[i] = sentence;

    if(adverbs == adverbUpper){
        adverbs = 203; //values are reset when upper bound is reached + next value is incremented
        verbs++;
    }
    if(verbs == verbUpper){
        verbs = 190;
        nouns++;
    }
    if(nouns == nounUpper){
        nouns = 1;
        names++;
    }
    adverbs++;
}
```

The hash values of the sentences were then put into an array, and a comparison algorithm was run on it. This method was run and approx. 4 hours in, a score of 22 was achieved. The algorithm also seemed to get high scores faster too.

Efficiency

The algorithm was rather slow, as initially the sentences were stored in the array and they were converted to hash as required. The sha256() method was called every time the sentences were referenced, and since each sentence would be compared multiple times, this was redundant. I decided to store the hash values of the sentences in the array instead of the sentences themselves. However, since there is no way to reverse the hash, there would be no reference to the original sentence. To accommodate for this and to allow the original sentence to be retrieved, the getSentence() method was created. The getSentence() method incorporated the same counter algorithm as the one that fills the array. The counter loop was run for an int ‘pos’, where pos is the position of the hash in the array.

Data Structures Used:

An array was used to store hash values. A queue was used

Algorithms Used:

A counter algorithm was used to generate sentences and retrieve sentences based on a position.

A comparison algorithm was used.

```
//comparison algorithm
for(int x = 0; x < hashVal.length - 1; x++){
    for(int y = x + 1; y < hashVal.length; y++){
        int score = calcScore(hashVal[x], hashVal[y]);
        if(score > maxScore && !(hashVal[x].equals(hashVal[y]))){
            maxScore = score;
            posX = x;
            posY = y;
            System.out.println(hashVal[x] + "\n" + hashVal[y] + " " + posX + " " + posY + "\n" + " " + maxScore);
            System.out.println(getSentence(posX) + "\n" + getSentence(posY) + "\n");

        }else if(score == maxScore){
            System.out.println("same score achieved. " + x + " " + y); //used to check for duplicate scores
        }
    }
    if(x % 1000 == 0) System.out.println("count = " + x); //used to keep track of where the algorithm is in the array
}
```

Conclusion:

This project was my first time working on a programme that ran for longer than a few seconds. It made me consider efficiency a lot more and resulted in a lot of iterations of code that performed the same function, but more efficiently. A score of 23 was achieved, unfortunately I edited the text file, making the positions of the pair invalid. The highest, with proof, was 22 with the sentence pair:

```
max positions were: 903885 978595
with a max score of: 22

the sentence pair was:
Jakobe's dress levitated speedily
Ishaan's cd floated promptly
a0ba93af8fc87b4db129714985c9c1578368d02dd592bf06c7b57df98e80d7f6
e0ba63afb3347054592c77798c91a95b536ebebb7734cb38c7953df28681d1c0
```

Code

```
import java.io.*;
import java.util.*;
import java.security.*;
public class assignmentiterativemethods {
   public static void main(String []args){

      Dictionary dic = new Dictionary();
      String first = "";
      String second = "";
      int nouns = 1;
      int nounUpper = 188;
      int verbs = 190;
      int verbUpper = 202; //indexes for the upper and lower bounds of each of the typed of words
      int adverbs = 204;
      int adverbUpper = 215;
      int names = 217;
      int namesUpper = 2129;

      String hashVal[] = new String [1000000];
      int count = 0;
      int maxScore = 0, posX = 0, posY = 0; //max score achieved so far, as well as the positions of the first and
second sentences in the best comparison

      /*counter algorithm, used to
        iterate through all possible
        sentences
      */
      for(int i = 0; i < hashVal.length ; i++){
         String name = dic.getWord(names);
         String sentence = name.charAt(name.length() - 1) == 's' ? name + "' " : name + "'s " + dic.getWord(nouns)
+ " " + dic.getWord(verbs) + " " + dic.getWord(adverbs) + ".";
         hashVal[i] = sentence;

         if(adverbs == adverbUpper){
            adverbs = 203; //values are reset when upper bound is reached + next value is incremented
            verbs++;
         }
         if(verbs == verbUpper){
            verbs = 190;
            nouns++;
         }
         if(nouns == nounUpper){
            nouns = 1;
            names++;
```

```
            }
            adverbs++;
        }
        //comparison algorithm
        for(int x = 0; x < hashVal.length - 1; x++){
            for(int y = x + 1; y < hashVal.length; y++){
                int score = calcScore(hashVal[x], hashVal[y]);
                if(score > maxScore && !(hashVal[x].equals(hashVal[y]))){
                    maxScore = score;
                    posX = x;
                    posY = y;
                    System.out.println(hashVal[x] + "\n" + hashVal[y] + " " + posX + " " + posY + "\n" + " " + maxScore);
                    System.out.println(getSentence(posX) + "\n" + getSentence(posY) + "\n");

                }else if(score == maxScore){
                    System.out.println("same score achieved. " + x + " " + y); //used to check for duplicate scores
                }
            }
            if(x % 1000 == 0) System.out.println("count = " + x); //used to keep track of where the algorithm is in the
array
        }
        System.out.println("max positions were: " + posX + " " + posY + "\n" + "with a max score of: " + maxScore);
        System.out.println("\n" + "the sentence pair was: " + "\n" + getSentence(posX) + "\n" +
getSentence(posY));
        System.out.println(hashVal[posX] + "\n" + hashVal[posY]);

    }
    //get sentence based on posX and posY
    public static String getSentence(int in){
        Dictionary dic = new Dictionary();
        int nouns = 1;
        int nounUpper = 188;
        int verbs = 190;
        int verbUpper = 202;
        int adverbs = 204;
        int adverbUpper = 215;
        int names = 217;
        int namesUpper = 2129;
        String sentence = "";

        for(int i = 0; i <= in; i++){
            sentence = dic.getWord(names) + "'s " + dic.getWord(nouns) + " " + dic.getWord(verbs) + " " +
dic.getWord(adverbs) + ".";

            if(adverbs == adverbUpper){
                adverbs = 203;
```

```
            verbs++;
        }
        if(verbs == verbUpper){
            verbs = 190;
            nouns++;
        }
        if(nouns == nounUpper){
            nouns = 1;
            names++;
        }
        adverbs++;
    }
    return sentence;
}
public static String sha256(String input){
    try{
        MessageDigest mDigest = MessageDigest.getInstance("SHA-256");
        byte[] salt = "CS210+".getBytes("UTF-8");
        mDigest.update(salt);
        byte[] data = mDigest.digest(input.getBytes("UTF-8"));
        StringBuffer sb = new StringBuffer();
        for (int i=0;i<data.length;i++){
            sb.append(Integer.toString((data[i]&0xff)+0x100,16).substring(1));
        }
        return sb.toString();
    }catch(Exception e){
        return(e.toString());
    }
}
//counts identical characters in two strings and returns an int
public static int calcScore(String in1, String in2){
    int score = 0;
    for(int i = 0; i < in1.length(); i++){
        if(in1.charAt(i) == in2.charAt(i))
          score++;
    }
    return score;
}
//dictionary class and setup
public static class Dictionary{

    private String input[];

    public Dictionary(){
        input = load("/Users/stephengallagher/Downloads/Untitled document.txt");
    }
```

```
    public int getSize(){
        return input.length;
    }

    public String getWord(int n){
        return input[n];
    }

    private String[] load(String file) {
        File aFile = new File(file);
        StringBuffer contents = new StringBuffer();
        BufferedReader input = null;
        try {
            input = new BufferedReader( new FileReader(aFile) );
            String line = null;
            int i = 0;
            while (( line = input.readLine()) != null){
                contents.append(line);
                i++;
                contents.append(System.getProperty("line.separator"));
            }
        }catch (FileNotFoundException ex){
            System.out.println("Can't find the file - are you sure the file is in this location: "+file);
            ex.printStackTrace();
        }catch (IOException ex){
            System.out.println("Input output exception while processing file");
            ex.printStackTrace();
        }finally{
            try {
                if (input!= null) {
                    input.close();
                }
            }catch (IOException ex){
                System.out.println("Input output exception while processing file");
                ex.printStackTrace();
            }
        }
        String[] array = contents.toString().split("\n");
        for(String s: array){
            s.trim();
        }
        return array;
    }
  }
}
```