Karandeep Jaswal

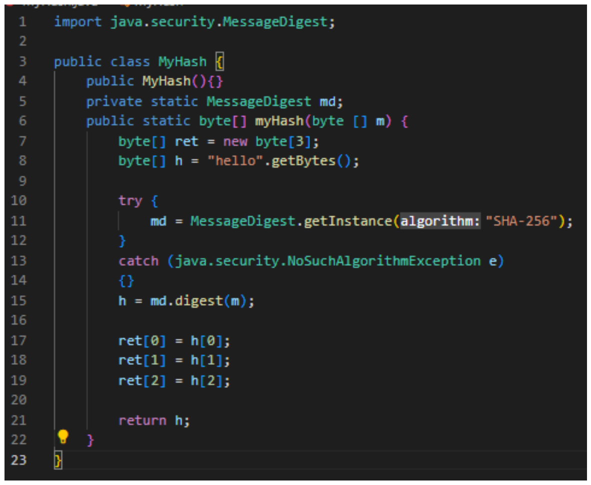
INCS 741

Assignment 1- Find Collision

10/31/22

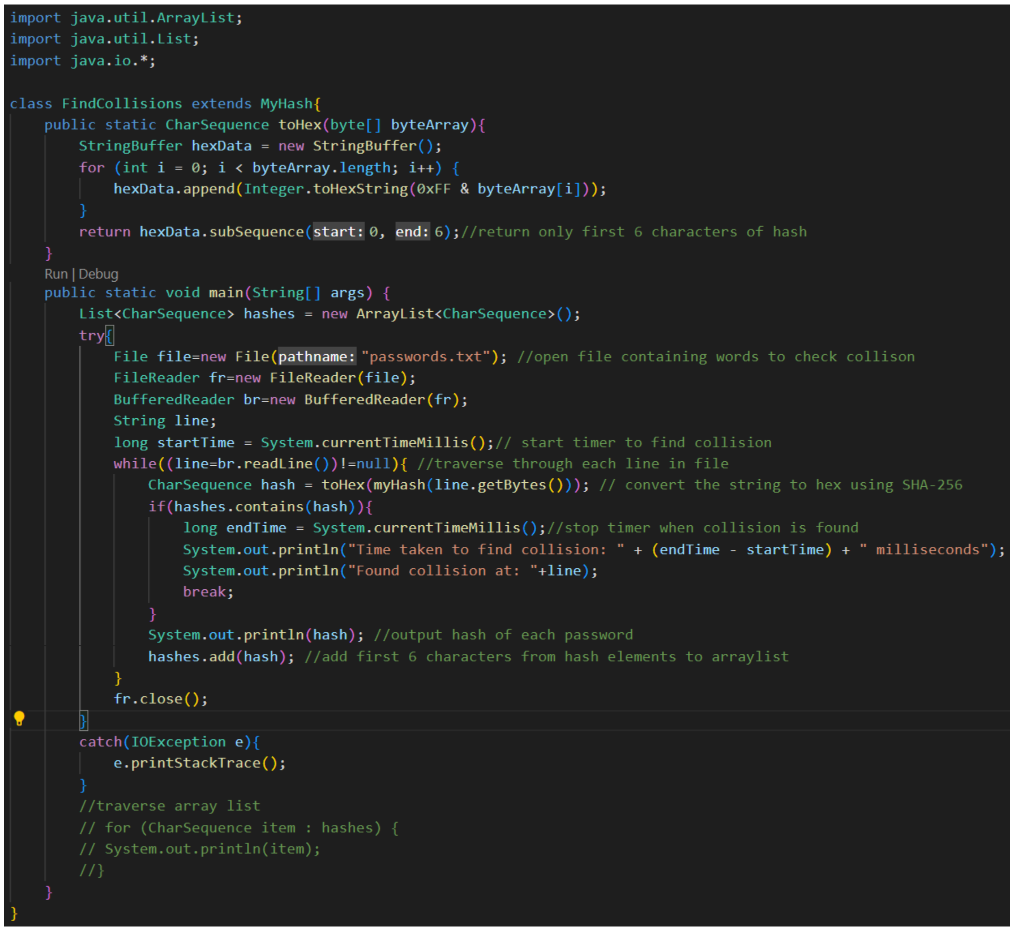
1. Introduction

The class **MyHash** outputs a hash of 24 bits (i.e., 3 bytes).



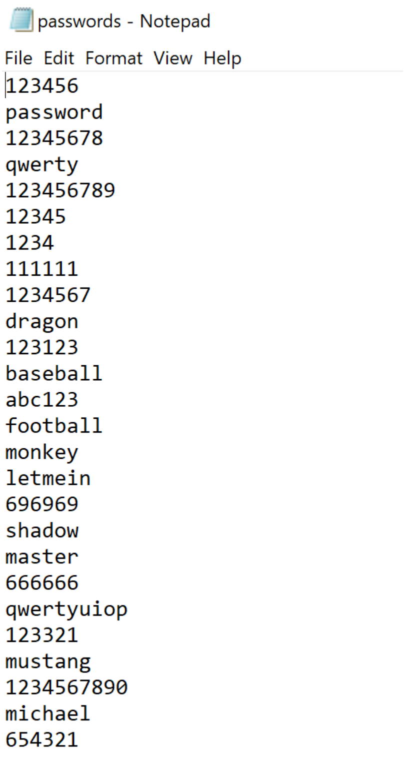
```java
import java.security.MessageDigest;

public class MyHash {
    public MyHash(){}
    private static MessageDigest md;
    public static byte[] myHash(byte [] m) {
        byte[] ret = new byte[3];
        byte[] h = "hello".getBytes();

        try {
            md = MessageDigest.getInstance(algorithm: "SHA-256");
        }
        catch (java.security.NoSuchAlgorithmException e)
        {}
        h = md.digest(m);

        ret[0] = h[0];
        ret[1] = h[1];
        ret[2] = h[2];

        return h;
    }
}
```

The class **FindCollisions** finds a collision and outputs the hash and the time it took the program to find a collision.



```java
import java.util.ArrayList;
import java.util.List;
import java.io.*;

class FindCollisions extends MyHash{
    public static CharSequence toHex(byte[] byteArray){
        StringBuffer hexData = new StringBuffer();
        for (int i = 0; i < byteArray.length; i++) {
            hexData.append(Integer.toHexString(0xFF & byteArray[i]));
        }
        return hexData.subSequence(start: 0, end: 6);//return only first 6 characters of hash
    }
    Run | Debug
    public static void main(String[] args) {
        List<CharSequence> hashes = new ArrayList<CharSequence>();
        try{
            File file=new File(pathname: "passwords.txt"); //open file containing words to check collison
            FileReader fr=new FileReader(file);
            BufferedReader br=new BufferedReader(fr);
            String line;
            long startTime = System.currentTimeMillis();// start timer to find collision
            while((line=br.readLine())!=null){ //traverse through each line in file
                CharSequence hash = toHex(myHash(line.getBytes())); // convert the string to hex using SHA-256
                if(hashes.contains(hash)){
                    long endTime = System.currentTimeMillis();//stop timer when collision is found
                    System.out.println("Time taken to find collision: " + (endTime - startTime) + " milliseconds");
                    System.out.println("Found collision at: "+line);
                    break;
                }
                System.out.println(hash); //output hash of each password
                hashes.add(hash); //add first 6 characters from hash elements to arraylist
            }
            fr.close();
        }
        catch(IOException e){
            e.printStackTrace();
        }
        //traverse array list
        // for (CharSequence item : hashes) {
        // System.out.println(item);
        //}
    }
}
```

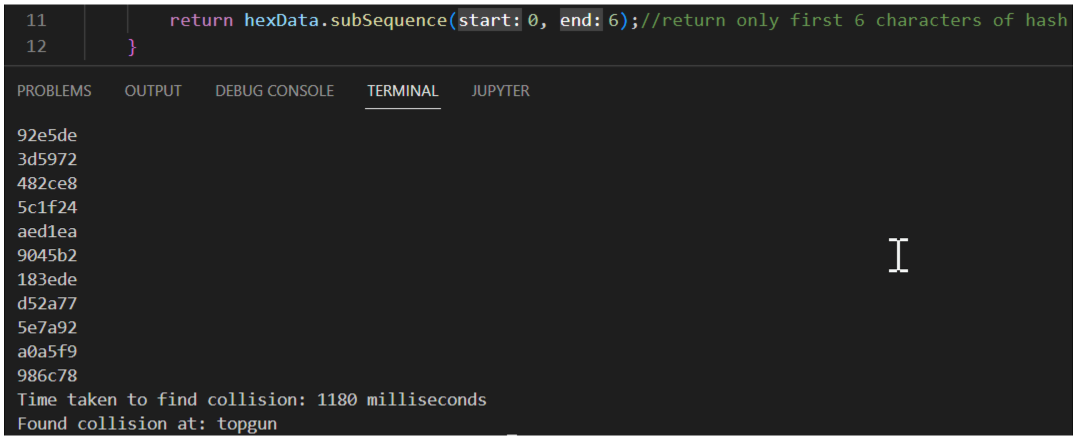
FindCollisions reads the **passwords.txt** file, which contains 471333 passwords in plaintext (please be advised some passwords contain profanity and vulgarity). **FindCollisions** with the use of **MyHash** hashes these passwords and finds a collision.



```
123456
password
12345678
qwerty
123456789
12345
1234
111111
1234567
dragon
123123
baseball
abc123
football
monkey
letmein
696969
shadow
master
666666
qwertyuiop
123321
mustang
1234567890
michael
654321
```

Finding Collision

By default, the program finds collision by comparing the first 6 characters (24 bits) of the hash. The collision is found in 1180 milliseconds.

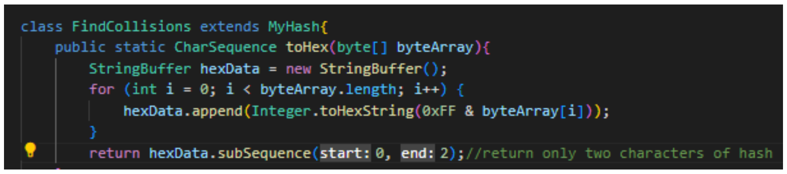


```
        return hexData.subSequence(start: 0, end: 6);//return only first 6 characters of hash
    }
```

```
92e5de
3d5972
482ce8
5c1f24
aed1ea
9045b2
183ede
d52a77
5e7a92
a0a5f9
986c78
Time taken to find collision: 1180 milliseconds
Found collision at: topgun
```

Note: Limitation- this time also includes the time taken to print the hashes.

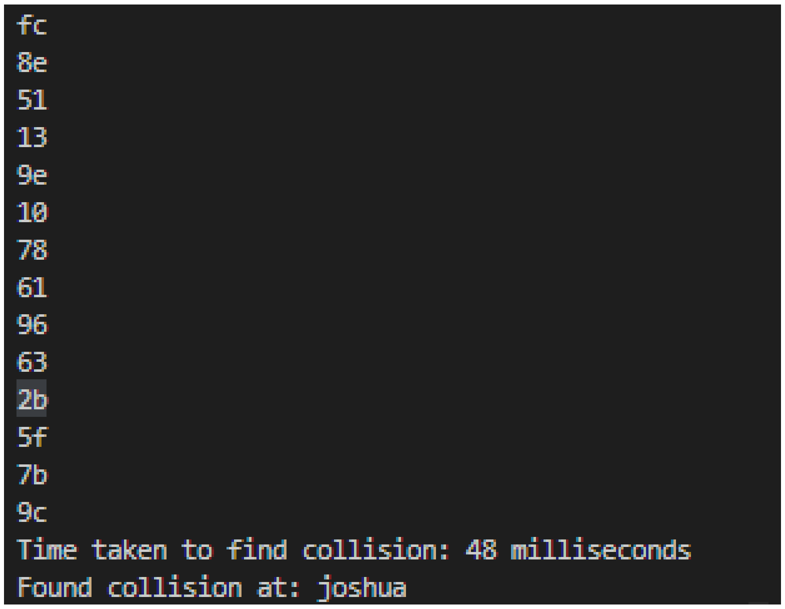
I. Finding collision, when using 8 bits of SHA-256



```
class FindCollisions extends MyHash{
    public static CharSequence toHex(byte[] byteArray){
        StringBuffer hexData = new StringBuffer();
        for (int i = 0; i < byteArray.length; i++) {
            hexData.append(Integer.toHexString(0xFF & byteArray[i]));
        }
        return hexData.subSequence(start: 0, end: 2);//return only two characters of hash
```

Changed the code to return hex data of 8 bits.

Output:

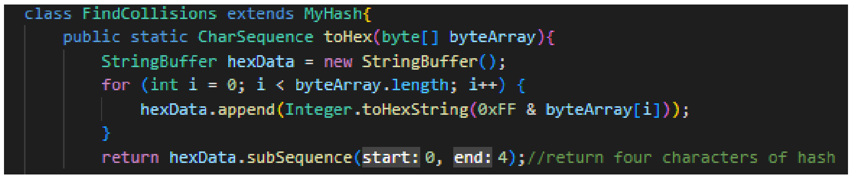


```
fc
8e
51
13
9e
10
78
61
96
63
2b
5f
7b
9c
Time taken to find collision: 48 milliseconds
Found collision at: joshua
```

2b hex = **00101011** Binary (8 bits)

After running the program 10 times, the average time taken to find a collision using 8 bits of SHA-256= (48+53+54+51+54+53+46+56+46+47)/ 10 milliseconds = **50.8 milliseconds.**

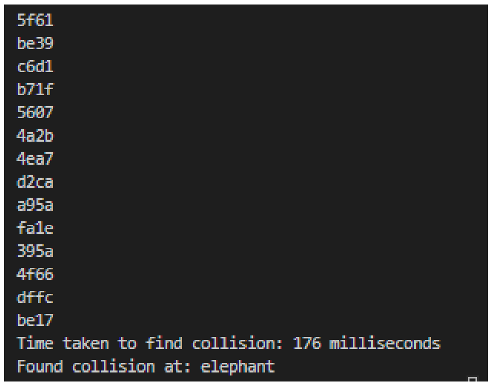
II. Finding collision, when using 16 bits of SHA-256



```
class FindCollisions extends MyHash{
    public static CharSequence toHex(byte[] byteArray){
        StringBuffer hexData = new StringBuffer();
        for (int i = 0; i < byteArray.length; i++) {
            hexData.append(Integer.toHexString(0xFF & byteArray[i]));
        }
        return hexData.subSequence(start: 0, end: 4);//return four characters of hash
```

Changed the code to return hex data of 16 bits.

Output:

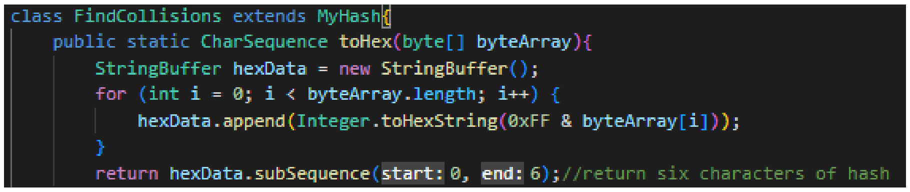


```
5f61
be39
c6d1
b71f
5607
4a2b
4ea7
d2ca
a95a
fa1e
395a
4f66
dffc
be17
Time taken to find collision: 176 milliseconds
Found collision at: elephant
```

be17 hex = **1011111000010111** Binary (16 bits)

After running the program 10 times, the average time taken to find a collision using 16 bits of SHA-256= (176+169+220+154+247+217+239+211+229+247)/ 10 milliseconds = **210.9 milliseconds.**

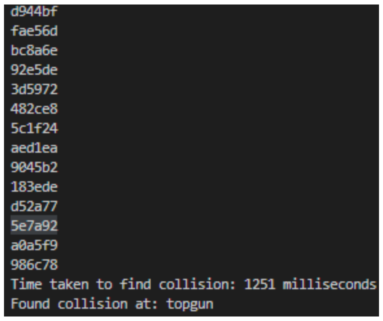
III. Finding collision, when using 24 bits of SHA-256



```
class FindCollisions extends MyHash{
    public static CharSequence toHex(byte[] byteArray){
        StringBuffer hexData = new StringBuffer();
        for (int i = 0; i < byteArray.length; i++) {
            hexData.append(Integer.toHexString(0xFF & byteArray[i]));
        }
        return hexData.subSequence(start: 0, end: 6);//return six characters of hash
```

Changed the code to return hex data of 24 bits.

Output:

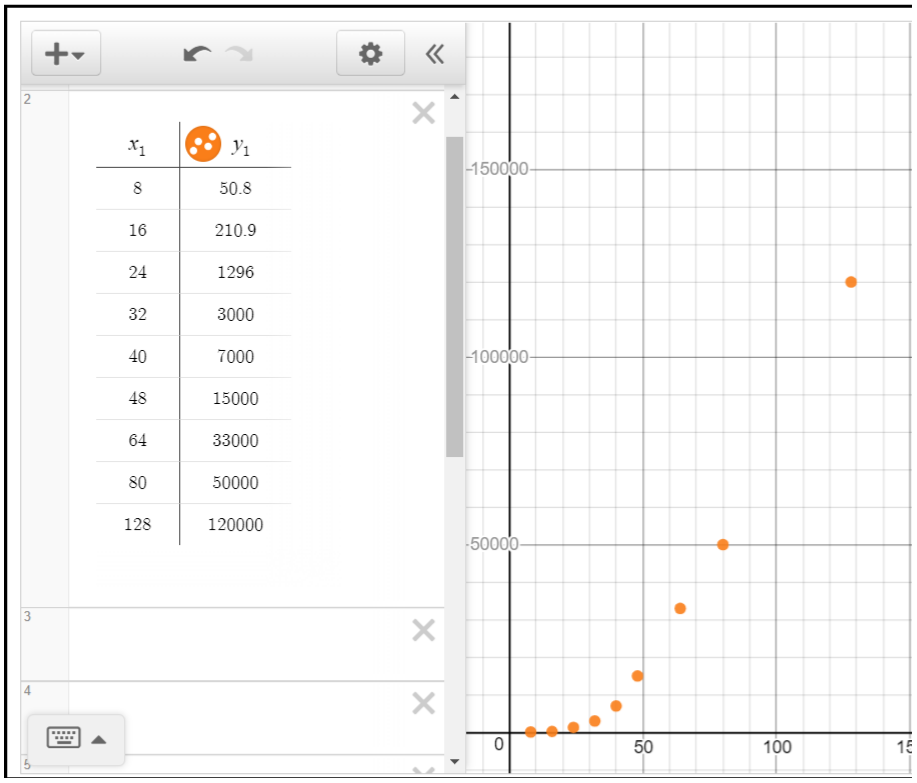


```
d944bf
fae56d
bc8a6e
92e5de
3d5972
482ce8
5c1f24
aed1ea
9045b2
183ede
d52a77
5e7a92
a0a5f9
986c78
Time taken to find collision: 1251 milliseconds
Found collision at: topgun
```

986c78 hex = **010111100111101010010010** binary (24 bits)

After running the program 10 times, the average time taken to find a collision using 24 bits of SHA-256= (1251+1316+1220+1360+1187+1403+1526+1314+1201+1182)/ 10 milliseconds = **1296 milliseconds**.

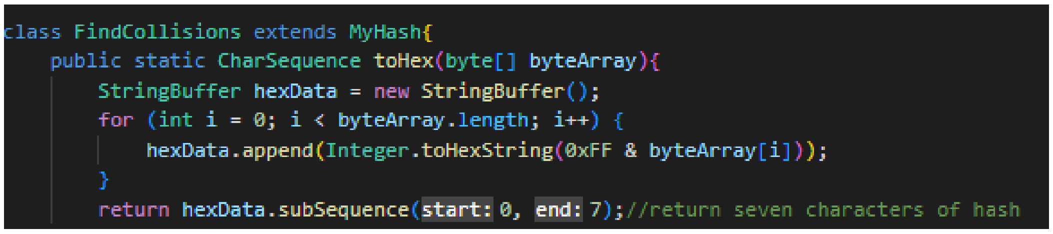
2. **Prediction Graph** – The graph grows exponentially. The x-axis represents the first n bits used for SHA-256 and the y-axis represents the time take to find a collision.



x_1	y_1
8	50.8
16	210.9
24	1296
32	3000
40	7000
48	15000
64	33000
80	50000
128	120000

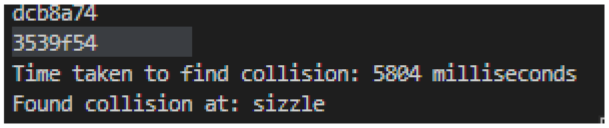
3. **Actual results**

I. Finding collision, when using 32 bits of SHA-256



```
class FindCollisions extends MyHash{
    public static CharSequence toHex(byte[] byteArray){
        StringBuffer hexData = new StringBuffer();
        for (int i = 0; i < byteArray.length; i++) {
            hexData.append(Integer.toHexString(0xFF & byteArray[i]));
        }
        return hexData.subSequence(start: 0, end: 7);//return seven characters of hash
```

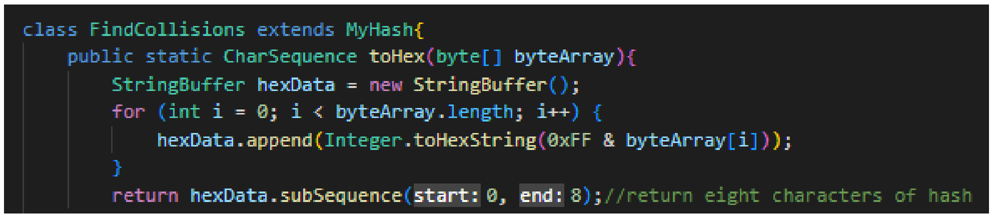
Output:



```
dcb8a74
3539f54
Time taken to find collision: 5804 milliseconds
Found collision at: sizzle
```

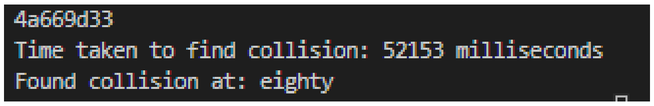
After running the program, the time taken to find a collision using 32 bits of SHA-256= 5804 milliseconds.

II. Finding collision, when using 40 bits of SHA-256



```java
class FindCollisions extends MyHash{
    public static CharSequence toHex(byte[] byteArray){
        StringBuffer hexData = new StringBuffer();
        for (int i = 0; i < byteArray.length; i++) {
            hexData.append(Integer.toHexString(0xFF & byteArray[i]));
        }
        return hexData.subSequence(start: 0, end: 8);//return eight characters of hash
```

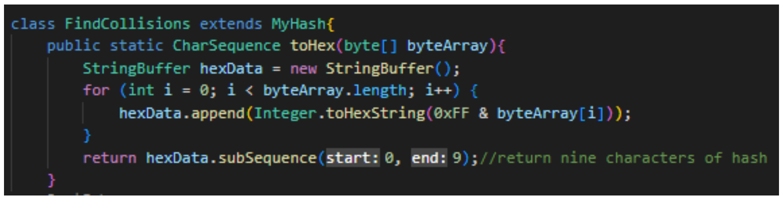
Output:



```
4a669d33
Time taken to find collision: 52153 milliseconds
Found collision at: eighty
```

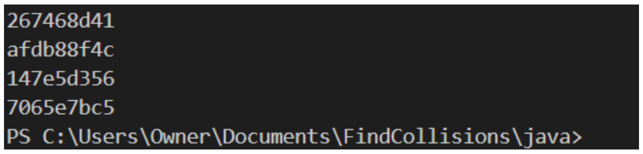
After running the program, the time taken to find a collision using 40 bits of SHA-256= 52153 milliseconds.

III. Finding collision, when using 48 bits of SHA-256



```java
class FindCollisions extends MyHash{
    public static CharSequence toHex(byte[] byteArray){
        StringBuffer hexData = new StringBuffer();
        for (int i = 0; i < byteArray.length; i++) {
            hexData.append(Integer.toHexString(0xFF & byteArray[i]));
        }
        return hexData.subSequence(start: 0, end: 9);//return nine characters of hash
    }
```

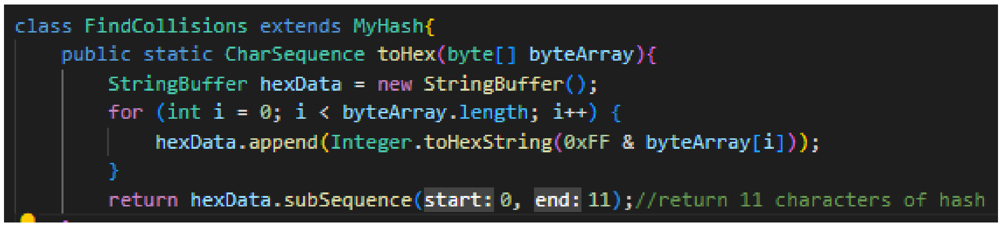
Output:



```
267468d41
afdb88f4c
147e5d356
7065e7bc5
PS C:\Users\Owner\Documents\FindCollisions\java>
```

After running the program, no collisions were detected.

IV. Finding collision, when using 64 bits of SHA-256

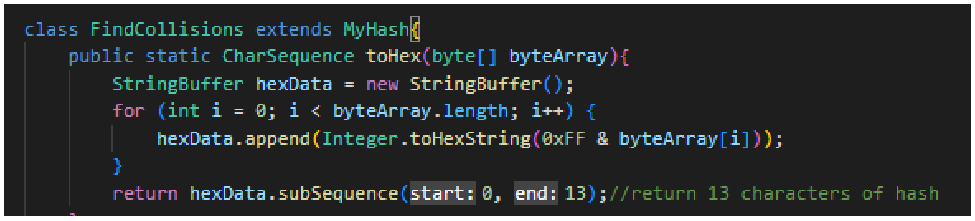


```
class FindCollisions extends MyHash{
    public static CharSequence toHex(byte[] byteArray){
        StringBuffer hexData = new StringBuffer();
        for (int i = 0; i < byteArray.length; i++) {
            hexData.append(Integer.toHexString(0xFF & byteArray[i]));
        }
        return hexData.subSequence(start: 0, end: 11);//return 11 characters of hash
```

Output:

After running the program, the program could not find a collision.

V. Finding collision, when using 80 bits of SHA-256



```
class FindCollisions extends MyHash{
    public static CharSequence toHex(byte[] byteArray){
        StringBuffer hexData = new StringBuffer();
        for (int i = 0; i < byteArray.length; i++) {
            hexData.append(Integer.toHexString(0xFF & byteArray[i]));
        }
        return hexData.subSequence(start: 0, end: 13);//return 13 characters of hash
```

Output:

After running the program, the program could not find a collision. The same was true for n=128 bits.

4. **Conclusion**

As the first n bits of SHA-256 hash get bigger, the more time it takes to find a collision. This was proved by finding collision time using the first n bits of SHA-256 for n ∈ {8, 16,24, 32,40, 48}.

When n,

8 – collision time = 50.8 ms

15– collision time= 210.9 ms

24– collision time= 1296 ms

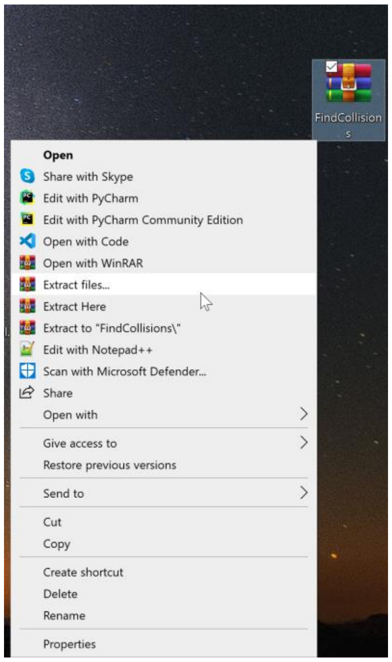
32– collision time= 5804 ms

40– collision time=52153 ms

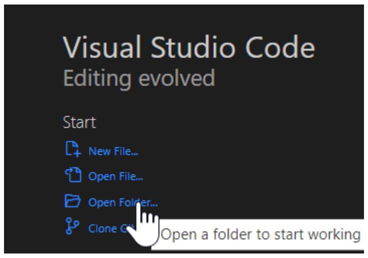
The prediction was significantly incorrect – slower than actual results. It can be noted that the graph grows extremely more exponentially than predicted, which illustrates the strength of SHA-256 hash function. My program has limited inputs (471333 passwords) to store as hashes. Eventually for n=48, 64, 80 and 128 bits, the program ran out of inputs that could be compared and no collision was detected after n=40 bits.

5. How to run the code

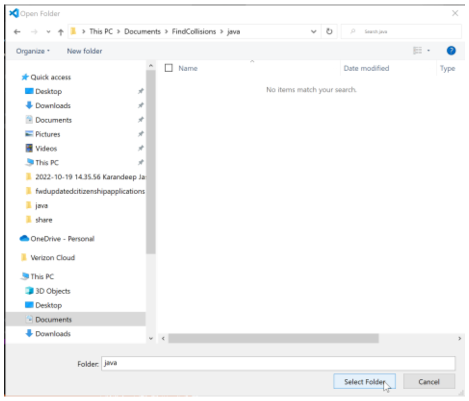
1) Download the FindCollisions.zip.
2) Unzip the file.



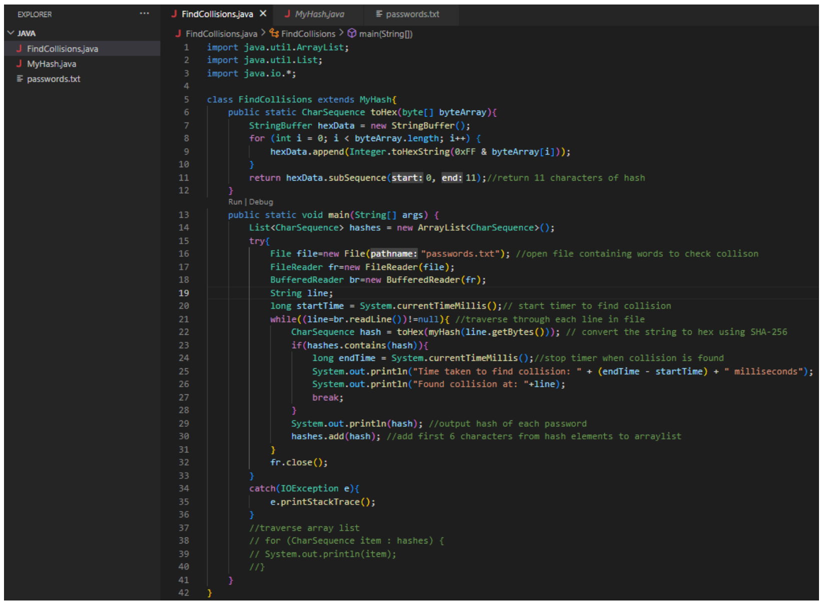
3) Open your code editor (I am using Visual Studio Code) and click on “Open Folder” (or equivalent).



4) Find the FindCollisions folder, then go to java and click “Select Folder”

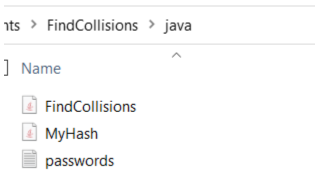


5) The 3 files (FindCollision.java, MyHash.java, and passwords.txt) should open.

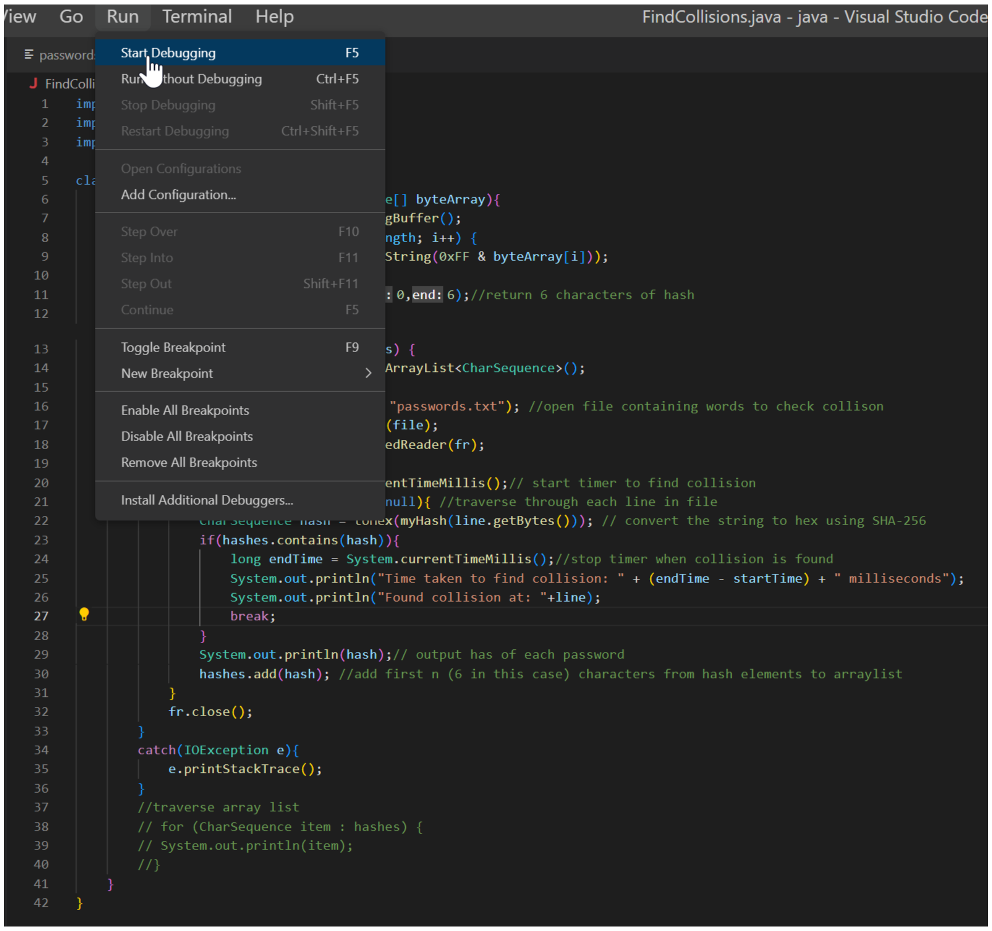


```java
import java.util.ArrayList;
import java.util.List;
import java.io.*;

class FindCollisions extends MyHash{
    public static CharSequence toHex(byte[] byteArray){
        StringBuffer hexData = new StringBuffer();
        for (int i = 0; i < byteArray.length; i++) {
            hexData.append(Integer.toHexString(0xFF & byteArray[i]));
        }
        return hexData.subSequence(start: 0, end: 11);//return 11 characters of hash
    }
    public static void main(String[] args) {
        List<CharSequence> hashes = new ArrayList<CharSequence>();
        try{
            File file=new File(pathname: "passwords.txt"); //open file containing words to check collison
            FileReader fr=new FileReader(file);
            BufferedReader br=new BufferedReader(fr);
            String line;
            long startTime = System.currentTimeMillis();// start timer to find collision
            while((line=br.readLine())!=null){ //traverse through each line in file
                CharSequence hash = toHex(myHash(line.getBytes())); // convert the string to hex using SHA-256
                if(hashes.contains(hash)){
                    long endTime = System.currentTimeMillis();//stop timer when collision is found
                    System.out.println("Time taken to find collision: " + (endTime - startTime) + " milliseconds");
                    System.out.println("Found collision at: "+line);
                    break;
                }
                System.out.println(hash); //output hash of each password
                hashes.add(hash); //add first 6 characters from hash elements to arraylist
            }
            fr.close();
        }
        catch(IOException e){
            e.printStackTrace();
        }
        //traverse array list
        // for (CharSequence item : hashes) {
        // System.out.println(item);
        //}
    }
}
```

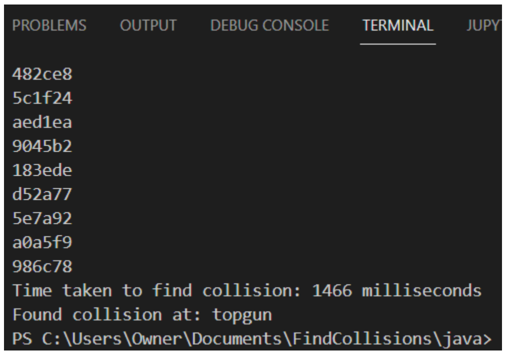
FYI, contents in the FindCollisions folder: there is a java folder which contains three files – FindCollisions.java, MyHash.java and passwords.txt.



6) On the FindCollisions.java file, go to Run> Start Debugging. After the Debugging process finishes, the code will execute.



```
imp
imp
imp

cla
                                   e[] byteArray){
                                   gBuffer();
                                   ngth; i++) {
                                   String(0xFF & byteArray[i]));

                                   : 0,end: 6);//return 6 characters of hash

                                   s) {
                                   ArrayList<CharSequence>();

                                    "passwords.txt"); //open file containing words to check collison
                                   (file);
                                   edReader(fr);

                                   entTimeMillis();// start timer to find collision
                                   null){ //traverse through each line in file
                CharSequence hash = toHex(myHash(line.getBytes())); // convert the string to hex using SHA-256
                if(hashes.contains(hash)){
                    long endTime = System.currentTimeMillis();//stop timer when collision is found
                    System.out.println("Time taken to find collision: " + (endTime - startTime) + " milliseconds");
                    System.out.println("Found collision at: "+line);
                    break;
                }
                System.out.println(hash);// output has of each password
                hashes.add(hash); //add first n (6 in this case) characters from hash elements to arraylist
            }
            fr.close();
        }
        catch(IOException e){
            e.printStackTrace();
        }
        //traverse array list
        // for (CharSequence item : hashes) {
        // System.out.println(item);
        //}
    }
}
```



```
482ce8
5c1f24
aed1ea
9045b2
183ede
d52a77
5e7a92
a0a5f9
986c78
Time taken to find collision: 1466 milliseconds
Found collision at: topgun
PS C:\Users\Owner\Documents\FindCollisions\java>
```