GTU Department of Computer Engineering
CSE 222/505 - Spring 2023
Homework #4 Report

REŞİT AYDIN

200104004019

1-Problem Definition

We are asked to design and the implement a security system for the museum in Topkapı Palace. This security system must provide an encrypted password for each museum officer. Our system is responsible for decoding the passwords to check if they are valid or not, if the passwords are valid, the system opens the entrance door to let the officer get in.

2-Problem Solution Approach

I created 3 separate classes named Password1, Password2 and Username. These classes contain methods related to their names to check their validity and structure. I also added additional helper functions to be used in main class or another method of a class. We had to design and use a recursive approach in the algorithm of some classes. To do that, first I developed an idea of implementing it with loops then converted it to a recursive function. I tried to keep my functions as time efficient as possible even if some of them are recursive which is not really efficient.

3- Time complexity (Big O) Analysis of the Functions

n – input size

boolean checkIfValidUsername(): In this recursive function, the best case occurs when the input size is 1, in this case best-case time complexity would be O(1) which is constant. In the worst-case scenario, where the input username consists of all digits, the function will recursively call itself for each character in the “username”, until it reaches the last character. This gives us a time complexity of O(n), where n is the length of the “username”.

boolean containsUserNameSpirit() : The time complexity of this method is O(n*m), where n is the length of” password1” and m is the length of “username”. This method creates a stack “pw” and pushes each character of “password1” onto the stack, which takes O(n) time, where n is the length of “password”. Next, the method enters a while loop that iterates as long as the stack “pw” is not empty. For each iteration of the loop, method iterates over each character in “username” and compares it to the top of the stack “pw” using the “peek” method. This comparison takes constant time. If a match if found the method immediately returns “true”. If not, the method pops out of the stack using “pop” method. This popping also takes constant time.

The worst-case scenario is when there is no common element between any character in “username” and “password1”, in which case the method will iterate over all the characters in both string which gives us a time complexity of O(n*m).

boolean isBalancedPassword() : The method first checks if “password1” is not null , which takes constant time. Then it creates three stack which each takes constant time. Next, it iterates over each character in “password1” using a for loop, which takes O(n) time. The method performs a constant time operations like push or pop for each character. In the worst-case scenario is when the “password1” consists of entirely opening parentheses, brackets etc. which takes O(n) time.

In general, the time complexity of this method is O(n).

boolean isPalindromePossible() : This is recursive method. For the first time this function is called, I call my helper removeBrackets() method to remove brackets from a string which takes O(n) time. In worst case, the method will be called recursively n times, where n is the length of the input string. In each recursive call, the function updates an array “alfabet” to keep track of the frequency of each character in the string using ASCII values of them. This operation takes O(n) time in the worst case.

The total time complexity of the method is O(n^2).

boolean isExactDivision() : This recursive function uses a while loop (which I couldn’t solve without using a n additional loop), to repeatedly subtract the largest possible denomination from “password” until it becomes zero. In the best case, “password2” is zero, and the function returns immediately. The time complexity of the function in the best case is O (1).

The while loop in the method runs at most m / denominations[n-1] times. So, in the worst case, the largest denomination is 1 and “m” is equal to “n”, which means that the while loop will run “m” times. Therefore, time complexity of the function is O (m) where m is the size of password2.

4- Test Cases and the Outputs

4.1- Test Cases

```
System.out.println("\n----------------TEST CASE 1---------------------");
testHelper(usr, pw1, pw2, username, password1, password2, denom);


System.out.println("\n----------------TEST CASE 2---------------------");
testHelper(usr, pw1, pw2, username:"sibelgulmez", password1:"[rac()ecar]", password2:74, denom);


System.out.println("\n----------------TEST CASE 3---------------------");
testHelper(usr, pw1, pw2, username:"", password1:"[rac()ecar]", password2:74, denom);


System.out.println("\n----------------TEST CASE 4---------------------");
testHelper(usr, pw1, pw2, username:"x123abc", password1:"[rac()ecar]", password2:74, denom);


System.out.println("\n----------------TEST CASE 5---------------------");
testHelper(usr, pw1, pw2, username:"ahmet", password1:"pass[]", password2:74, denom);


System.out.println("\n----------------TEST CASE 6---------------------");
testHelper(usr, pw1, pw2, username:"ahmet", password1:"abcdabcd", password2:74, denom);

System.out.println("\n----------------TEST CASE 7---------------------");
testHelper(usr, pw1, pw2, username:"ahmet", password1:"[[[()]]]", password2:74, denom);


System.out.println("\n----------------TEST CASE 8---------------------");
testHelper(usr, pw1, pw2, username:"ahmet", password1:"[hya](xyz)", password2:74, denom);


System.out.println("\n----------------TEST CASE 9---------------------");
testHelper(usr, pw1, pw2, username:"ahmet", password1:"[axhs{})", password2:74, denom);


System.out.println("\n----------------TEST CASE 10---------------------");
testHelper(usr, pw1, pw2, username:"ahmet", password1:"[r{sxayr}s]", password2:74, denom);


System.out.println("\n----------------TEST CASE 11---------------------");
testHelper(usr, pw1, pw2, username:"ahmet", password1:"[rac()ecar]", password2:28, denom);

System.out.println("\n----------------TEST CASE 12---------------------");
testHelper(usr, pw1, pw2, username:"newuser", password1:"{abeccbaxx()}", password2:54, denom);
```

4.2- Outputs

```
resitaga@resitaga:~/Desktop/HW4$ javac Test.java
resitaga@resitaga:~/Desktop/HW4$ java Test

-----------------TEST CASE 1--------------------
The username and passwords are valid. The door is opening, please wait...

-----------------TEST CASE 2--------------------
The username and passwords are valid. The door is opening, please wait...

-----------------TEST CASE 3--------------------
The username is invalid. It should have letters only and it can't be empty.

-----------------TEST CASE 4--------------------
The username is invalid. It should have letters only and it can't be empty.

-----------------TEST CASE 5--------------------
The password1 is invalid. It should have at least 8 characters and 2 brackets. It should at least have one letter.

-----------------TEST CASE 6--------------------
The password1 is invalid. It should have at least 8 characters and 2 brackets. It should at least have one letter.

-----------------TEST CASE 7--------------------
The password1 is invalid. It should have at least 8 characters and 2 brackets. It should at least have one letter.

-----------------TEST CASE 8--------------------
The password1 is invalid. It should be possible to obtain a palindrome from the password1.

-----------------TEST CASE 9--------------------
The password1 is invalid. It should be balanced.

-----------------TEST CASE 10--------------------
The password1 is invalid. It should be possible to obtain a palindrome from the password1.

-----------------TEST CASE 11--------------------
The password2 is invalid. It is not compatible with the denominations.

-----------------TEST CASE 12--------------------
The username and passwords are valid. The door is opening, please wait...
resitaga@resitaga:~/Desktop/HW4$
```

In Test class I created a testHelper () method to print out different outputs with respect to the return values from the boolean functions. This function takes Username, Password1, Password2 objects and String username, String password1, int password2 and int [] denominations as parameters. In this method, it calls the functions related to username , password1 and password2 then prints statements according to the return values. In main method, I created Username, Password1, Password2 objects and Strings for password1 and username, int for password2 and int [] for denominations. After that, I called testHelper () function by sending these objects as parameters.