ITCS231

Data Structures and Algorithm Analysis

Programming Project 1

Waris Damkham 6388014

Section 1

Dr. Pawitra Liamruk

Linear time: O(n)

Code:

```java
import java.util.Scanner;
public class Linear {
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number :");
        double num = sc.nextDouble();
        sc.close();
        System.out.println("Starting Attacking......");
        for (int i = 0; i < num; i++)
        {
            System.out.println("Hacking Yours Computer " +i+"%");
        }
        System.out.println(".........................");
        System.out.println("Hacking Yours Computer 100%");
        System.out.println("Yours Computer Hacked Successfully :D ");
    }
}
```

This algorithm seeks information based on the amount of data available or user input if the number of steps required to complete the performance of an algorithm should write the loop increases or decreases significantly when the number of inputs.

This program is a program that runs following an input to print out "Hacking Yours Computer input number," and after running the program, the program will print "Hacking Yours Computer 100%" and "Your Computer Hacked Successfully :D."

The Output:

```
Enter number :23
Starting Attacking......
Hacking Yours Computer 0%
Hacking Yours Computer 1%
Hacking Yours Computer 2%
Hacking Yours Computer 3%
Hacking Yours Computer 4%
Hacking Yours Computer 5%
Hacking Yours Computer 6%
Hacking Yours Computer 7%
Hacking Yours Computer 8%
Hacking Yours Computer 9%
Hacking Yours Computer 10%
Hacking Yours Computer 11%
Hacking Yours Computer 12%
Hacking Yours Computer 13%
Hacking Yours Computer 14%
Hacking Yours Computer 15%
Hacking Yours Computer 16%
Hacking Yours Computer 17%
Hacking Yours Computer 18%
Hacking Yours Computer 19%
Hacking Yours Computer 20%
Hacking Yours Computer 21%
Hacking Yours Computer 22%
.........................
Hacking Yours Computer 100%
Yours Computer Hacked Successfully :D
```

Constant time: O(1)

Code:

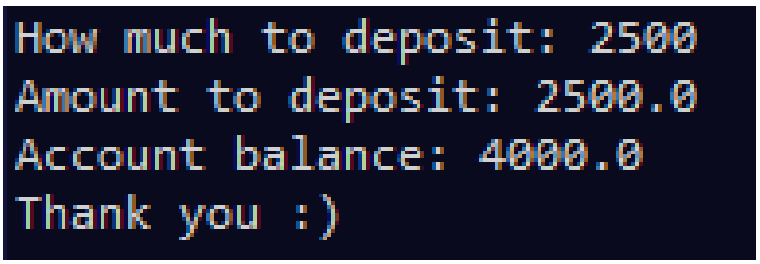
```java
import java.util.Scanner;
public class ConstantTime {
    public static void main(String[] args)
    {
        int balance = 1500;
        Scanner sc= new Scanner(System.in);
        System.out.print("How much to deposit: ");
        double deposit = sc.nextInt();
        System.out.println("Amount to deposit: " + deposit);
        if(deposit<=100)
        {
            System.out.println("Sorry this is too low for a deposit; try again");
            System.out.println("============================================");
            System.out.print("How much to deposit: ");
            double deposit2 = sc.nextInt();
            sc.close();
            System.out.println("Amount to deposit: " + deposit2);
            if(deposit2<=100)
            {
                System.out.println("Please run the program again ");
                System.out.println("============================================");
                System.exit(0);
            }
            else
            {
                System.out.println("Account balance: " + (deposit2+balance));
                System.out.println("Thank you :)");
            }
        }
        else
        {
            System.out.println("Account balance: " + (deposit+balance));
            System.out.println("Thank you :)");
        }
    }
}

```

This algorithm is the most straightforward because no matter how large the amount of information, the Processing time will take only one cycle.

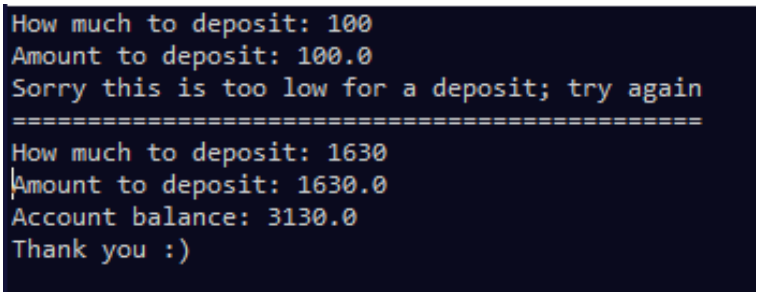
This program is a program in which users input an amount of money to deposit. However, if the amount of money is too low, the program will ask again to enter more money to reach the minimum for the deposit process, but if a customer enters that not reach the minimum, please rerun the program, but if the customer exceeds the minimum the program print out all balance in a customer bank account.

The Output:

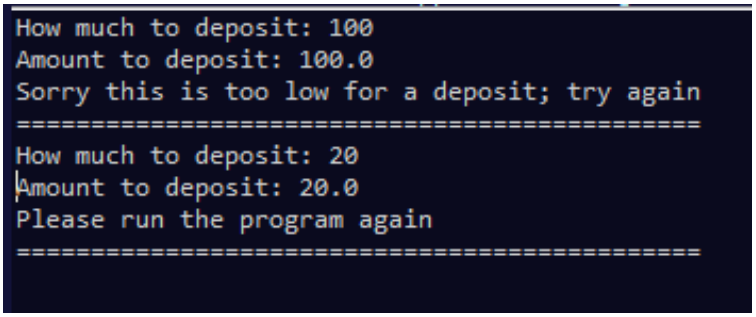
1. If the amount of deposit exceed the minimum in the first input.



2. If the amount of deposits exceed the minimum in the second input.



3. If the amount of deposits does not reach a minimum both times.



Quadratic time: O(n2)

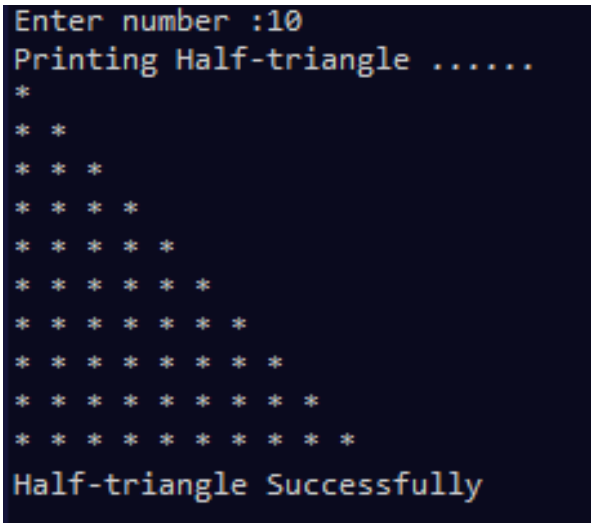
Code:

```java
import java.util.Scanner;
public class Quadratic {
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number :");
        double num = sc.nextDouble();
        sc.close();
        System.out.println("Printing Half-triangle ......");
        for(int i=0; i < num; i++)
        {
            for(int j =0 ;j<=i;j++)
            {
                System.out.print("* ");
            }
        System.out.println();
        }
    System.out.println("Half-triangle Successfully");
    }
}
```

This code is a two-nested loop, also known as loop nesting. When the steps needed to perform the algorithm are a quadratic function of the number of items in the input, the algorithm is quadratic. The symbol O(n2) represents quadratic complexity.

This program is a program that runs following an input to print out a Half-triangle pattern.

The Output:



Reference

Big O Notation and Algorithm Analysis with Python Examples. (2021, September 1). Stack Abuse. https://stackabuse.com/big-o-notation-and-algorithm-analysis-with-python-examples/

Big O Notation. (n.d.). Cooervo. Retrieved September 11, 2021, from https://cooervo.github.io/Algorithms-DataStructures-BigONotation/big-O-notation.html