greatlearning
Power Ahead

Problem Solving in Programming

Table of Content

- Problem Introduction
- Flow Chart of the problem
- Pseudocode of the problem
- Algorithm of the problem
- Implementation of the problem

Problem Introduction

- Majority of problems are solvable by computers
- Few things to consider:
  - Analysis
  - Number and types of inputs
  - Output and its type
  - Constraints
  - Flow chart
  - Pseudocode
  - Algorithm

greatlearning
Power Ahead

Problem Statement - Modified Armstrong Number

A number num is considered to be modified armstrong if it is equal to the sum of digits raised to the power of total number of digits in num and it doesn't end with 0.

For example :153
As 1+125+27 = 153

Your task is check if the given number is armstrong number or not.

greatlearning
Power Ahead

Analysis

- A number is given
- Need to check if given number is armstrong number
- Questions
  - Ask for range of the number
  - Ask for the type of output expected
  - Ask if the total number of digits is given or not

I/O

- Input
  - Type : Integer
  - Number of input : 1

- Output
  - Type : Boolean (Assumed)
  - Number of output : 1

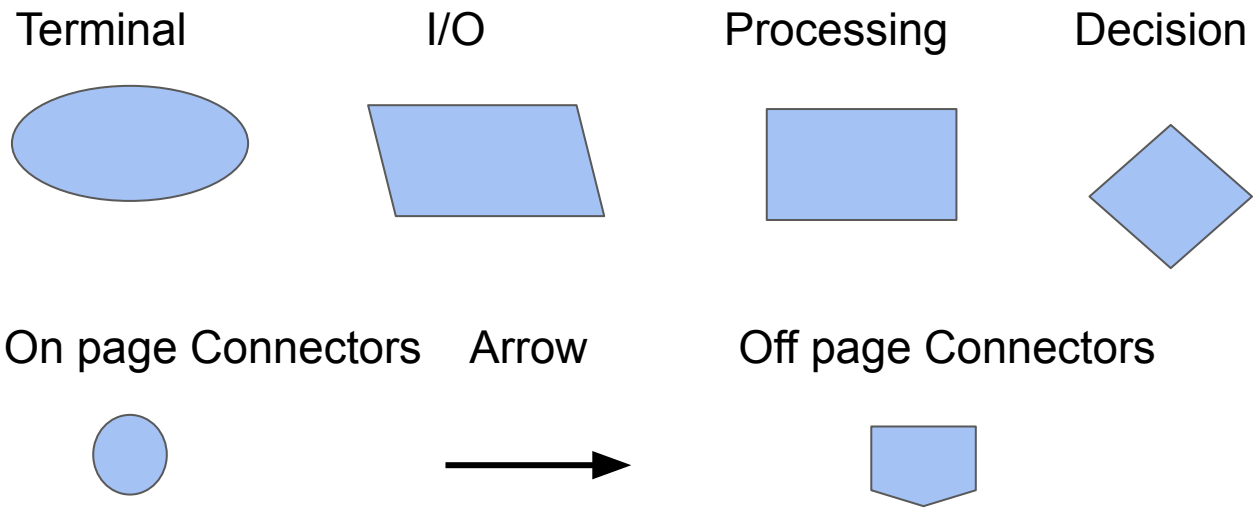
Constraints

- Number must not end with 0
- Conditions of armstrong number
- Number of digits is not given(assumed)

Flow Chart

- Pictorial representation of steps to be performed
- Enable visualization of the problem
- Shows clear data flow with the help of arrows
- Can be used with non technical audience too
- Have a definite start and end point

Flow Chart - Symbols



greatlearning
Power Ahead

Flow Chart - Modified Armstrong Number

Pseudocode

- Pseudocode is step by step approach of a problem in simple english
- It can be easily understood by layman and follow no programming construct
- It basically maps the flowchart in simple statements
- Serves in documentation purpose which is vital in organizations

greatlearning
Power Ahead

Pseudocode - Modified Armstrong Number

- Input a number
- Check the last digit of the number. If it is 0 then return false.
- Count the total number of digits in a number
- Take each digit and add digit raised to the power total number of digits. Add it to the sum variable.
- Check if the number is equal to the sum obtained. If yes then return true else return false.

greatlearning
Power Ahead

Algorithm

- It’s is more close to actual programming
- It follows programming construct to some extent
- Algorithm is independent of any programming language
- It is used to analyze time space complexity
- It helps in further optimization of code

Algorithm - Modified Armstrong Number

- Input num
- if num%10==0 return false
- tmp=num
- count=0;
- while tmp>0
  - count++
  - tmp=tmp/10
- tmp=num
- sum=0

Algorithm - Modified Armstrong Number

- while tmp>0
  - sum=sum+Math.pow(tmp%10,count)
  - tmp=tmp/10
- if sum==num return true
- else return false

QUESTION

greatlearning
Power Ahead

Find the element that is present once in an array where every other element is present twice?

Input:[1,3,5,6,6,3,1]
Output: 5

Input:[10,30,50,60,60,30,10]
Output: 50

greatlearning
Power Ahead

First Approach

Using NESTED LOOPS: Time Complexity: O(n*n)

```
for(int i=0;i<arr.length;i++) {
      for(int j=0;j<arr.length;j++) {
            if(i != j && arr[i]==arr[j])
               flag=1;
        }
         if(flag==0){
              System.out.println(arr[i]);
              break;
           }
        flag=0;
 }
```

greatlearning
Power Ahead

Second Approach

Using SORTING: Time Complexity: O(n*log n)

```
Arrays.sort(arr);
        for(int i=0;i<arr.length;i=i+2)
            {
                if(arr[i]!=arr[i+1])
                  {
                    System.out.println(arr[i]);
                    break;
                  }
            }
```

Third Approach

greatlearning
Power Ahead

Using HASHMAP: Time Complexity: O(n) Space Complexity: O(n)

```
HashMap<Integer,Integer> map = new HashMap<Integer,Integer>();
        for(int i=0;i<arr.length;i++) {
                if(map.containsKey(arr[i]))
                    map.put(arr[i],map.get(arr[i])+1);
                 else
                    map.put(arr[i],1);
            }
            System.out.println(map);  //FOR DEBUGGING
              for(int x :map.keySet()) {
               if(map.get(x)==1){
                        System.out.println(x);
                           break;
                    }
            }
```

Fourth Approach

Using for loop: Time Complexity: O(n) Space Complexity: O(1)

```
int res=0;
        for(int i=0;i<arr.length;i++) {
            {
                res= res^ arr[i];

            }
         System.out.println(res);
```

Bitwise XOR

INPUT X	INPUT Y	OUTPUT
0	0	0
0	1	1
1	0	1
1	1	0

So, 8 ^8 = 0 And 8^0 = 8
That is what we used: 1^1 ^ 3^3 ^ 5 ^ 6^6 = 5^0 = 5

greatlearning
Power Ahead

Thank You