Structure and Analysis of Algorithm

Assignment-1

Group-2 (IEC2021005-IEC2021009)

Introduction to Programming

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, ALLAHABAD

PROFESSOR:- DR.MUHAMMAD JAVED SIR

Contents:-

Problem Statement

Given an unsorted array of n numbers, find the sum of all the perfect numbers and check whether it is Armstrong number or not.

Introduction

- Given an array arr[] of size n, we will find the sum of all the perfect numbers and further we will check whether the sum is an Armstrong number or not.

- A **perfect number** is a positive integer that is equal to the sum of all its proper positive divisors excluding the number itself. For e.g.: 6,28, 256, etc. 6 is a perfect number because the proper divisor of 6 are 1,2,3 (excluding 6) and sum of all its positive divisors (1+2+3) is 6.

- An **Armstrong number** of order n is a number in which each digit when multiplied by itself n number of times and finally added together, results the same number. For eg: 371 is a three-digit number. Therefore, order=3. now, here each digit is multiplied by itself three times and finally added together and results in our original number. For e.g.:
  (3*3*3+7*7*7+1*1*1) = (27+343+1) = 371

Algorithm Design

- We will take the size of array and input elements in it.Then we will check whether each element is perfect number or not. Using for loop and if statement, we will find the divisors and if its sum comes out to be number itself, it will be a perfect number. In this way we will store the sum of all these perfect numbers(=d) and check whether it is an Armstrong number or not.

- For this we will calculate the number of digits(=nod) in d using while loop and extract each digit of d(=e) so as to calculate the sum of digits raised to power of nod. If this sum is equal to d then it will be an Armstrong Number otherwise not.

Algorithm Illustration

Begin

int b

Bool Value of perfect=FALSE

Array a[100]

DISPLAY- "Enter number of elements"

INPUT b

DISPLAY- "Enter elements"

for (int i=0; i<b; i=i+1)

INPUT a[i]

EndFor

int sum, d;

for (i=0; i<b; i=i+1)

sum=0,d=0;

for (int j=1; j <= a[i] / 2; j=j+1)

if (a[i]%j==0) then

sum = sum + j

EndIf

EndFor

if (sum==a[i]) then

DISPLAY- "(sum) is a perfect number"

d=d+sum AND

Bool Value of perfect=TRUE

EndIf

EndFor

if (**Bool Value** of perfect==**TRUE**) then

DISPLAY- "d is the sum of perfect numbers"

int sum1=d, digsum=0, nod=0

while (sum1! = 0) then

nod=nod+1 AND

sum1=(sum1 /10)

EndWhile

Int sum2=d

while (sum2 != 0) then

int e=sum%10 AND

digsum = digsum + pow(e, nod)

sum2=sum2/10

EndWhile

if (digsum==d) then

DISPLAY- "d is an Armstrong number"

EndIf

else

DISPLAY- "d is not an Armstrong number"

EndElse

EndIf

else then

DISPLAY- "There is no perfect number"

EndElse

Time Complexity

- The Average Execution time of above algorithm is calculated as 0.00481375 second. It is calculated using following syntax:

```
#include <time.h>

clock_t start, end;
double cpu_time_used;

start = clock();
... /* Do the work. */
end = clock();
cpu_time_used = ((double) (end - start)) / CLOCKS_PER_SEC;
```

Conclusion

- With the help of this problem we may conclude that it becomes easy to find perfect number and Armstrong number using while loop. Instead of re-writing the code again and again we only have to iterate using while loop till the condition is true.

- We may also conclude that it becomes easy to store many inputs in a single variable using array. Otherwise for each input value, we may have to assign a new variable.

References

- https://www.geeksforgeeks.org/program-for-armstrong-numbers/
- https://www.geeksforgeeks.org/perfect-number/
- https://www.geeksforgeeks.org/sum-factors-number/
- "Let us C"- book by Yashavant Kanetkar