**Experiment Title 1.1**

**Student Name: Sahul Kumar Parida UID: 20BCS4919**

**Branch: CSE Section/Group: WM-904/B**

**Semester: 5th Date of Performance: 21/08/2022**

**Subject Name: Design and Analysis Algorithm Lab**

**Subject Code: 20CSP-312**

**1. Aim/Overview of the practical:**

Calculate the greatest common divisor (GCD) of two numbers using code and analysis.

**2. Task to be done/ Which logistics used:**

Using Brute Force Approach.

Using Euclid Algorithm Approach.

**3. Algorithm/Flowchart (For programming based labs):**

1. Brute Force Algorithm: -

* Find smaller of two numbers.
* Iterate from 2 to smaller number.
* If the number is divisible by both the numbers store it in the array.

1. Efficient Approach Euclid Algorithm: -

* Find which number is greater at first.
* Use a recursive stack call until the smaller number becomes 0.
* Return greater number If smaller number becomes 0 i.e. the base case.
* Else we need to call GCD using smaller number and the modulo of greater and smaller number.

1. We can also do the above approach iteratively.

**4. Steps for experiment/practical/Code:**

#include <iostream>

using namespace std;

void solve(int a, int b)

{

if(b==0)

{

cout<<"Using Recursive Euclid Approach GCD is:- " << a;

return;

}

solve(b, a%b);

}

int main()

{

cout<< "SAHUL KUMAR PARIDA"<<endl;

cout<< "20BCS4919"<<endl;

int a,b;

cout << "Enter first number: ";

cin >> a;

cout << "Enter second number: ";

cin >> b;

int ans;

// BRUTE FORCE APPROACH

// Time Complexity: O(N)

for(int i=1; i<=min(a,b);i++)

{

if(a%i==0 && b%i==0)

{

ans=i;

}

}

cout<<"Using Brute Force Approach GCD is:- " << ans<<endl;

int a2=a;

int b2=b;

//ITERATIVE EUCLID APPROACH

//Time Complexity: O(LOG N)

//Space Complexity: O(1)

while(b!=0)

{

int c=a;

a=b;

b=c%b;

}

cout<< "Using Iterative Euclid Approach GCD is:- "<< a<<endl;

//RECURSIVE APPROACH

//Time Complexity: O(LOG N)

//Space Complexity: O(LOG N)

solve(max(a,b),min(a,b));

}

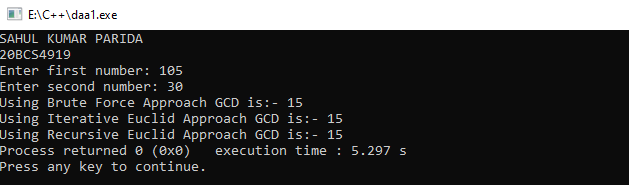
**5. Observations/Discussions/ Complexity Analysis:**

**Brute Force Approach: -** We iterate through the numbers until we find the maximum GCD the worst case time complexity i.e. Big O is O(min(a, b)). We have not used any sort of space, so space complexity is O(1).

**Recursive Euclid Approach:** - The time complexity of this solution is O(log(min(a, b))) which can be proved by the derivation. The space complexity of this solution is also the same i.e. O(log(min(a, b))) which is used in the recursive call stack.

**Iterative Euclid Approach: -** The time complexity of this solution is also the same i.e. O(log(min(a, b))) but since there is no recursive call stack. Hence, the space used is O(1). This is by far the most efficient solution.

**6. Result/Output/Writing Summary:**

****

**Learning outcomes (What I have learnt):**

1. To implement problems based on different algorithm design techniques.

2. To learn the importance of designing an algorithm in an effective way by considering space and time complexity.

3. Analyse and compare the efficiency of algorithms (Here we have compared brute force and both Iterating Method and Recursive method for finding the GCD).

4. Here Euclidean iterative algorithm proved to be more efficient to solve the problem for finding the GCD.