**Experiment Title 1.2**

**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:**

The code implements a power function in O(logn) time.

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

Using Efficient Algorithm.

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

1. Efficient Approach Euclid Algorithm: -

* Make an answer variable and initialize it to 1.
* Recursively solve the question while power doesn’t become equal to 1.
* Multiply the number by number itself and store in number.
* Divide the power by 2.
* If power is not divisible by 2 store the value in answer by multiplying answer with the number.

1. We can also do the above approach iteratively.

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

Recursive Approach

#include <bits/stdc++.h>

using namespace std;

//RECURSIVE APPROACH

//Time Complexity: O(log|n|)

//Space Complexity: O(log|n|)

float power(float x, int y)

{

if(y == 0)

return 1;

if (y % 2 == 0)

return power(x, y / 2) \* power(x, y / 2);

else

{

if(y > 0)

return x \* power(x, y / 2) \* power(x, y / 2);

else

return (power(x, y / 2) \* power(x, y / 2)) / x;

}

}

int main()

{

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

cout<< "20BCS4919"<<endl;

float x;

int y;

cout << "Enter number: ";

cin>>x;

cout << "Enter power: ";

cin>>y;

cout << "Using Recursive Approach:-"<<power(x, y);

return 0;

}

Iterative Approach

#include <bits/stdc++.h>

using namespace std;

//ITERATIVE APPROACH

//Time Complexity: O(log|n|)

//Space Complexity: O(1)

long power(int x, unsigned n)

{

// initialize result by 1

long pow = 1L;

// multiply `x` exactly `n` times

for (int i = 0; i < n; i++) {

pow = pow \* x;

}

return pow;

}

int main()

{

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

cout<< "20BCS4919"<<endl;

int x;

unsigned n;

cout << "Enter number: ";

cin>>x;

cout << "Enter power: ";

cin>>n;

cout << "Using Iterative Approach:-"<< power(x, n);

return 0;

}

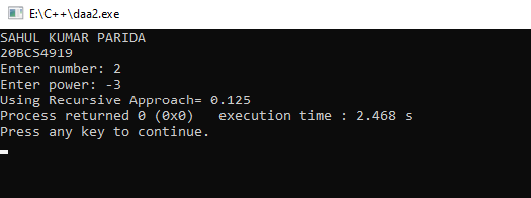
**5. Observations/Discussions/ Complexity Analysis:**

The Recursive approach has a time complexity of O(log N) since the operations are minimized by division with 2. It has a space complexity of O(log N) taken as the recursive stack space.

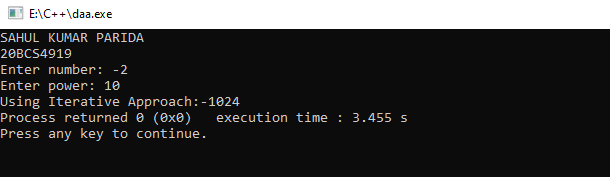
The Iterative approach has same time complexity as in Recursive approach i.e. O(log N) but the space complexity is O(1) since no space is used in a loop.

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

Recursive Approach



Iterative Approach

****

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

1. Using exponential model in C++.

2. Finding power of number.

3. Using different approaches for same problem and identifying the optimized approach.

4.Concepts of time and space complexity.