CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

NAAC GRADE A+
Accredited University

EXPERIMENT – 2

Student Name: Gaganjot Singh **UID: 22BCS14843**
Branch: BE-CSE **Section/Group:** 22BCS_JT_802-B
Semester: 5th **Date of Performance:** 25 July 2024
Subject Name: Design &Analysis of Algorithms **Subject Code:** 22CSH-311

1. **Aim:** Implement power function in O(Log(n)) time complexity.

2. **Objective:** The implement power function in O(Log(n)) time complexity so that we can solve power of any number with reduced time complexity. Usually, the n^2 is used when we apply it simply looping to solve it. Here we are using recursion to reduce the time complexity.

3. **Algorithm**
   - Start
   - Take Input.
   - Prompt user to enter the base number.
   - Prompt the user to enter the exponent.
   - Function definition : power(x,y)
   - If y==0, return 1 (base case, any number raised to 0 is 1)
   - Call power(x,y/2) and store the result in temp.
   - If y is even, return temp * temp
   - If y is odd, return x*temp*temp.
   - Output: Print the result of power (x,y) with beautiful formatting.
   - End

4. Implementation/Code:

```
#include <iostream>
using namespace std;
int power(int x, int y)
{
   if (y == 0)
   {
      return 1;
   }

   int temp = power(x, y / 2);
   if (y % 2 == 0)
   {
      return temp * temp;
   }
   else
   {
      return x * temp * temp;
   }
}
int main()
{
   int num, exp;
   cout << "Enter the number whose power you want to find:\n";
   cin >> num;
   cout << "Enter the power you want to find:\n";
   cin >> exp;
   cout << "The result is: " << power(num, exp);
   return 0;
}
```

5. Output

```
PS E:\CU Study\22CSH 311 DAA> cd "e:\CU Study\22CSH 311 DAA\" ; if ($?) { g++ ex2
.cpp -o ex2 } ; if ($?) { .\ex2 }
Enter the number whose power you want to find:
4
Enter the power you want to find:
3
The result is: 64
```

6. Time Complexity :

- The time complexity of the power functions implemented using recursion is O(log(n)).
- The function repeatedly, divides the exponent by 2 and squares the base until the exponent becomes 0. The number of iterations required to reach the base case if Log(n), where log(n) denotes the logarithm to the base 2. Therefore, the time complexity of the function is proportional to the number of iterations which is O(Log(n)).

7. Learning Outcomes:

- Implemented recursive functions
- Used RF to break down problem into smaller problems
- Divide and conquer approach for computation
- Recognize the importance of addressing the special cases, such as exponent zero.