Deliverable 1

Gurjot Singh

Student ID: 40091208

Problem 1:

Give a brief description, not exceeding one page, of your function, including the domain and co-domain of function, and the characteristics that make it unique.

The function $f(x,y) = x^y$ is also called power function in the field of Mathematics. It refers to raising x to the power of y. Here, x is the base and y is the exponent to which x is raised to.[2]

DOMAIN

All real numbers.

CO-DOMAIN

All real numbers.

Characteristics of the function

1. As the power increases, the graph of power function flattens somewhat near the origin and become steeper away from the origin.
2. The function is **not a one to one** function. For Example:- both 2^2 and -2^2 evaluates to the same result (4).
3. However, this function is an **onto function**.

Problem 2:

Express requirements of your function based on the style given in the ISO/IEC/IEEE 29148 Standard. Associate each requirement with a unique identifier.

Functional requirements for function : $f(x,y) = x^y$

First Requirement

ID - R1

Type - Functional Requirement

Version - 1.0

Difficulty - Intermediate

Description - The function calculates the result and displays in appropriate numeric format.

Second Requirement

ID - R2

Type - Functional Requirement

Version - 1.0

Difficulty - Easy

Description - If the inputs are provided in a format other than numeric values, it should throw an error.

Third Requirement

ID - R3

Type - Functional Requirement

Version - 1.0

Difficulty - Easy

Description - The function validates the input values. These should fall within the domain of the function (i.e. Real numbers).

Fourth Requirement

ID - R4

Type - Functional Requirement

Version - 1.0

Difficulty - Easy

Description - If the value of y is a decimal number, the value of x should be 0 or greater than 0.

Problem 3:

Give a brief description of your algorithms and express each of them in pseudocode.

Below are the two algorithms which were considered for calculating the power function.

Algorithm A: It involves an recursive approach for calculating the powers of integer values of x and y. The recursion is further made smart by checking whether the power is even or odd, and finding the power accordingly, in $O(log(y))$ time, which is very efficient with respect to the approach used in algorithm B. For calculating decimal powers of a number, this algorithm breaks the problem into two sub-problems:-

1. Calculating the power of **Integral-Part** of y .
2. Calculating the power of **Fractional-Part** of y .

Problem 1 is solved as described above, but for problem 2, the decimal number is first converted into fraction. The denominator part in this fraction denotes the Nth root of x. After calculating the Nth root of x, the Nth root is then raised to the Numerator of that fraction.

After both these sub-problems are solved, the results of problem 1 and 2 are multiplied together and we get our answer.

Advantage of Algorithm A:

Its amortized time complexity is $O(log(y))$. Hence, it is very efficient with respect to the normal way of finding a power by basic multiplication. Moreover, recursion is easy to understand and has high readability.

Disadvantage of Algorithm A:

It is implementation is bit complex.

Algorithm B: It involves the conventional approach for calculating the powers by using iterative multiplication.[1] Its time complexity is $O(y)$, which is not very efficient with respect to the approach used in algorithm A.

Advantage of Algorithm B:

It is pretty straightforward to implement as compared to the algorithm A. Also, it avoids memory overflow of input.

Disadvantage of Algorithm B:

Its time complexity is $O(y)$. Therefore, for large values of y, it takes a lot of time to execute. Hence, it is not that efficient as compared to the algorithm A.

After considering the advantages and disadvantages of both A and B, I selected the **algorithm A**.

Pseudocode for Algorithm A

Calculate: $f(x,y) = x^y$

Algorithm 1 POWER(x,y)

0: **IsADecimalNumber(a):** It checks whether a is not an integer.

IsAnInteger(a): It checks whether a is an integer.

GetNR(a): It returns the Numerator of the decimal number(a) after converting it into fraction.

GetDR(a): It returns the Denominator of the decimal number(a) after converting it into fraction.

GetNthPower(a,n): It returns the nth root of integer a.

1. $Input(x, y)$
2. if $x = 0$ then
3.   if $y \neq 0$ then
4.   return 0
5.   else
6.   return UNDEFINED
7.   end if
8. end if
9. if $y = 0$ then
10.   if $x \neq 0$ then
11.   return 1
12.   else
13.   return UNDEFINED
14.   end if
15. end if
16. if $x < 0$ then
17.   if $IsADecimalNumber(y)$ then
18.   return ERROR
19.   end if
20. end if
21. if $IsAnInteger(y)$ then
22.   return $POWER(x, y)$
23. else
24. $num \leftarrow GetNR(DecimalPart(y))$
25. $den \leftarrow GetDR(DecimalPart(y))$
26. $NthRoot \leftarrow GetNthPower(x, denominator)$
27.   return $POWER(x, IntegralPart(y)) * POWER(NthRoot, num)$
28. end if

Pseudocode for Algorithm B

Calculate: $f(x,y) = x^y$

Algorithm 2 POWER

1. $Input(x,y)$
2. $output \leftarrow 1$
3. while $y \neq 0$ then
4. $\quad output \leftarrow output * x$
5. $\quad y \leftarrow y - 1$
6. end while
7. return output

References

[1] Programiz. *Iterative Approach for power function calculation*. URL: `www.programiz.com`.

[2] Wikipedia. *Power Function*. URL: `https://en.wikipedia.org/wiki/PowerFunction`.