Question#2:

Part1:

Algorithm detail:

Given a and n

Answer = 1

For i in range(n):

Answer = Answer \* a

Return Answer

Algorithm idea:

Using the divide and conquer algorithm, each time I split the exponent into 2. With inputs of a(base number) and n(power), in order to calculate a^n in O(log(n)) times, I can create a function call pow(base, exponent). In this function, first check the exponent is equal to 0 or not, If 0, return 1. Else, divide the exponent by 2 and call pow(a,exponent/2) and get the value to variable z. Then check the current exponent whether is even or not, if exponent is even, I can return z\*z. else mean there the exponent cant divide properly, so need to return current base\*z\*z. when recursion done, I will get my final a^n in O(log(n)) times since I keep splitting the exponent in half.

Algorithm detail:

pow(base,exponent){

if exponent == 0: # base case

return 1;

z = pow(base,exponent/2); #split the exponent in half and call function

if exponent%2==0:

return z\*z

else:

return base\*z\*z #extra base because of the odd exponent

}

# function call with input a,n:

answer = pow(a,n)

return answer

Proof of Correctness idea:

Each use of pow function will divide exponent in half unless exponent equal to 0. Therefore, it will need to take log(n) times to reach the bottom of the divided tree. And the algorithm above also checks the case where the exponent can not be divided evenly, in that case, the number return will need to multiply z\*z by an additional base due to unevenly divided exponent. With this algorithm, it will go to all the way bottom of the tree and recursive back to the top level. Therefore, in the end after recursion done, I will get my a^n in O(log(n)) times.

Runtime Analysis:

The program will start going back when reach deepest level of the tree and since each time I split the exponent in half each time. Therefore, it need to take log(n) times to reach the deepest level. Therefore, the overall runtime of this algorithm is O(log(n)) times.

Source:

Lecture Note

Divide and Conquer Algorithm