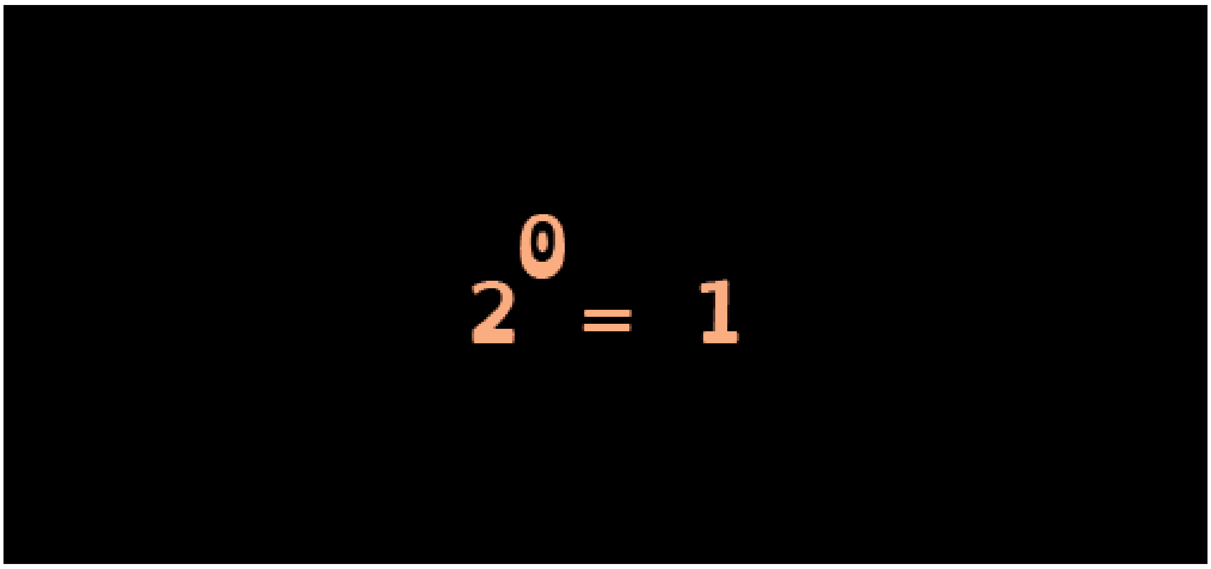

Binary Exponentiation

Tags	Algorithm Divide&Conquer MyNotes classic problem
Created time	@November 13, 2022 8:11 AM
Created time 1	@November 13, 2022 8:11 AM
Last edited time	@November 14, 2022 3:21 AM
Status	In progress
URL	
URL 1	

Binary Exponentiation is an algorithm to calculate power of a number in O(log n) complexity. instead of linear time using brute force.

by the help of some basic mathematics properties like:

Property of associativity (X * Y) * Z = X * (Y * Z)

Property of exponents (law of product) X^P = X^n * X^m (if n+m = p)

Algorithm - the first rule of optimisation of any algorithm is to take the brute force method and see if we can divide the problem in small sub problem and see are there some steps that we are repeating and then optimise accordingly.

so the brute force method for this problem will be loop from 0 to n-1 and multiplying x to itself multiple time

```
long long pow(int num,int exp){
  int ans = 1;
  for(int i = 0;i < exp;i++){
    ans *= num;
  }
  return ans;
}
```

now lets divide this into smaller problems

we know if we take 3 ^ 12 then it can be taken as 3^6 * 3^6 and we get the same answer

so now our objective is to divide the exponent x into it numbers whose sum is x itself and so the best way to divide the exponent is to divide it in multiple of twos because getting the multiple of twos will be easy using the binary representation of x and calculating will also will be easy since we will just have to square the answer variable instead of multiplying the number again and again.

example - 3^12

first lets see the binary representation of

firstly divide 12 in multiple of twos using bits = > 12(base-10) <==> 1 1 0 0 (base-2)

2^0 * 0 + 2^1 * 0 + 2^3 * 1 + 2^4 * 1 == 2^3 + 2^4 == 4 + 8

(one more observation from this - we are taking number only if the bit is “on” and it can be useful in program in if conditions)

so now we can write write 3^12 as 3^4 * 3^8

we have divided it but it is still is not that simplified we still have to calculate 3^4 and 3^8 separately

(or not) this is were the multiple of two comes handy we don’t have to calculate the power of 4 and 8 separately we can just take power of 4 and square it we will get power of 8 and now the question must be how to calculate power 4? just square the number before it which is 2 in this case and that can make our answer incorrect since we dont have to ad 3^2 so what we will do is keep a seperate variable for storing answer check if the bit is on or not if it is then multiply it on the go square the number then right shift the exponent.

```
long long power(int num,int exp){
   long long ans = 1;
   while(exp){
    ans = ((exp&1)?ans*num:ans);
    num = num * num;
    exp = exp >> 1;
   }
   return ans;
}
```

problem related to this comes in coding interview

▼ modular exponentiation 🔗

```
int modularExponentiation(int x, int n, int m) {
  // Write your code here.
    long long answer = 1;
    while(n){
        if(n&1){
            answer = ((answer%m) * 1ll * (x%m)) % m;
        }
        x = ((x%m) * 1ll * (x%m)) % m;
        n>>=1;
    }
    return answer;
}
```