Pow(x, n)

Implement pow(x, n).

Solution 1

```
public class Solution {
    public double pow(double x, int n) {
        if(n == 0)
            return 1;
        if(n<0){
            n = -n;
            x = 1/x;
        }
        return (n%2 == 0) ? pow(x*x, n/2) : x*pow(x*x, n/2);
    }
}
```

written by pei+heng original link here

Solution 2

/* This is a simple solution based on divide and conquer */

```
public class Solution {
       public double pow(double x, int m) {
           double temp=x;
           if(m==0)
           return 1;
           temp=pow(x,m/2);
           if(m%2==0)
           return temp*temp;
           else
           {
           if(m > 0)
               return x*temp*temp;
           else
               return (temp*temp)/x;
           }

   }
```

written by mohit4 original link here

Solution 3

After reading some good sharing solutions, I'd like to show them together. You can see different ideas in the code.

1. nest myPow

```
double myPow(double x, int n) {
    if(n<0) return 1/x * myPow(1/x, -(n+1));
    if(n==0) return 1;
    if(n==2) return x*x;
    if(n%2==0) return myPow( myPow(x, n/2), 2);
    else return x*myPow( myPow(x, n/2), 2);
}
```

2. double myPow

```
double myPow(double x, int n) {
    if(n==0) return 1;
    double t = myPow(x,n/2);
    if(n%2) return n<0 ? 1/x*t*t : x*t*t;
    else return t*t;
}
```

3. double x

```
double myPow(double x, int n) {
    if(n==0) return 1;
    if(n<0){
        n = -n;
        x = 1/x;
    }
    return n%2==0 ? myPow(x*x, n/2) : x*myPow(x*x, n/2);
}
```

4. iterative one

```
double myPow(double x, int n) {
    if(n==0) return 1;
    if(n<0) {
        n = -n;
        x = 1/x;
    }
    double ans = 1;
    while(n>0){
        if(n&1) ans *= x;
        x *= x;
        n >>= 1;
    }
    return ans;
}
```

5. bit operation

see this solution

If you have other ideas, please leave it below. Thanks.

written by mingjun original link here