GeeksforGeeks

A computer science portal for geeks

GeeksQuiz

- Home
- Algorithms
- DS
- GATE
- Interview Corner
- Q&A
- C
- C++
- Java
- Books
- Contribute
- Ask a Q
- About

Array
Bit Magic
C/C++
Articles
GFacts
Linked List
MCQ
Misc
Output
String
Tree
Graph

Write a C program to calculate pow(x,n)

Below solution divides the problem into subproblems of size y/2 and call the subproblems recursively.

```
#include<stdio.h>

/* Function to calculate x raised to the power y */
int power(int x, unsigned int y)
{
    if( y == 0)
        return 1;
    else if (y%2 == 0)
        return power(x, y/2)*power(x, y/2);
    else
        return x*power(x, y/2)*power(x, y/2);
```

```
}

/* Program to test function power */
int main()
{
    int x = 2;
    unsigned int y = 3;

    printf("%d", power(x, y));
    getchar();
    return 0;
}
```

Time Complexity: O(n)
Space Complexity: O(1)
Algorithmic Paradigm: Divide and conquer.

Above function can be optimized to O(logn) by calculating power(x, y/2) only once and storing it.

```
/* Function to calculate x raised to the power y in O(logn)*/
int power(int x, unsigned int y)
{
    int temp;
    if( y == 0)
        return 1;
    temp = power(x, y/2);
    if (y%2 == 0)
        return temp*temp;
    else
        return x*temp*temp;
}
```

Time Complexity of optimized solution: O(logn)
Let us extend the pow function to work for negative y and float x.

```
/* Extended version of power function that can work
 for float x and negative y*/
#include<stdio.h>

float power(float x, int y)
{
    float temp;
    if( y == 0)
       return 1;
    temp = power(x, y/2);
    if (y%2 == 0)
        return temp*temp;
    else
    {
        if(y > 0)
            return x*temp*temp;
        else
            return (temp*temp)/x;
    }
}

/* Program to test function power */
int main()
```

```
{
    float x = 2;
    int y = -3;
    printf("%f", power(x, y));
    getchar();
    return 0;
}
```

Related Topics:

- Generic Linked List in C
- Bit Fields in C
- Write your own memcpy()
- Some Interesting facts about default arguments in C++
- Comparison of a float with a value in C
- Pure virtual destructor in C++
- C++ mutable keyword
- Is it possible to call constructor and destructor explicitly?

Tags: Divide and Conquer

Writing code in comment? Please use **ideone.com** and share the link here.

The basic idea is to use bit manipulation

Reply • Share ›

grecodes • 7 months ago

what is the logic they are using here for negative y .I didn't understand the extended version code

Reply • Share ›

Ekta Goel ➚ grecodes • 6 months ago

a^(-1) is 1/a. This is the only thing they have done here i.e. if power is negative then we need to reciprocate our result.

Reply • Share ›

grecodes • 7 months ago

What is the logic they are using in the extended version of power function that can work for float x and negative y

Reply • Share ›

grecodes • 7 months ago

can someone explain the complexity of the code ? How is the complexity in the first case is O(n)
The master theorem equation should be T(n)=2*T(n/2)+Theta(1).

1 Reply • Share ›

Anurag Singh ➚ grecodes • 7 months ago

Master_theorem This is Case 1 scenario
a = 2, b = 2 and c = 0

c (=0) < logba (=1)
So T(n) = O(n^logba) = O(n^1) = O(n)

1 Reply • Share ›

Ravjot Singh • 9 months ago

I think the time complexity given in the article should be O(nlogn) rather O(n). Since this is divide and conquer approach and the equation will be T(n) = 2(T(n/2) + theta(1).
Please verify for the same

2 Reply • Share ›

abhinav ➚ Ravjot Singh • 9 months ago

no if you solve this T(n) = 2*T(n/2) + 1 you will get complexity O(n)
T(n) = 2*(T(n/4)+1) +1 = 1 + 2 + 4 ... logn terms so
1(2^logn - 1)/2-1 (sum of gp) = n so O(n)

and if we store one part then T(n) = T(n/2) + 1 = 1 + 1 + T(n/4)
=> 1 + 1 + 1 ..... logn times = so O(logn)

Reply • Share ›

Ravjot Singh ➚ abhinav • 9 months ago

My bad error in my calc. was it took logab =0 which was n :P Thanks :)

Reply • Share ›

Guest ➚ abhinav • 9 months ago

My bad I wrote a wrong equation in the first comment.

The equation can't be T(n) = 2T(n/2) + theta(1) it will be

T(n) = T(n/2) + theta(1) because the division is done only once not twice

So using maters theorem this will be case 2 since f(n) will be equal to n^(log base 2 1) = 1

hence it will be T(n) = n^(log base 2 1)* logn = logn

Reply • Share ›

h_gen • 10 months ago

Here is another non-recursive implement.

```
int power(int x, int n )
{
int r = 1;
while( n != 0 )
{
if (n % 2 != 0 )
r = x * r;
x *= x;
n /= 2;
}
return r;
}
```

The time-complexity is O(logN) also!

2 Reply • Share ›

grecode ➚ h_gen • 7 months ago

you missed to paste the link

http://ideone.com/5peNpQ

Reply • Share ›

Hitesh ➚ h_gen • 8 months ago

Good one. Can you explain the complexity.

▲ | ▼ • Reply • Share ›

h_gen ➚ Hitesh • 8 months ago

The complexity was dominated by the loop. n will be divided to 2 each time -> it takes log n times and the complexity, therefore, will be O(log n)

▲ | ▼ • Reply • Share ›

Deepesh Panjabi • a year ago

http://ideone.com/EUt4FE

▲ | ▼ • Reply • Share ›

Ajay Gaur • a year ago

Can someone tell me how to compute time complexity of the given algorithm?

▲ | ▼ • Reply • Share ›

learner • a year ago

http://ideone.com/NELonh in O(nlogn)

▲ | ▼ • Reply • Share ›

learner • a year ago

http://ideone.com/1gxsk2

▲ | ▼ • Reply • Share ›

kaushik Lele • a year ago

If same problem is to be solved in Java; it can be done easily using bitwise operator.

```
public class Try {

        public static int power(int num, int power){
if(power==0)
return 1;

                if(power==1)
return num;

//   Bitwise left shift operator viz. <<
                // n^2 = n <<1
// n^3 = n <<2
// n^4 = n <<3
// if power is even then n^power = n<<power -1="" if="" power="" is="" odd="" then="" n^pow
}
```

see more

• Reply • Share ›

HAppyHunting ➚ kaushik Lele • a year ago
only works for power of 2's ??
1 • Reply • Share ›

Renu ➚ HAppyHunting • 10 months ago
yes it will work only for power of 2
• Reply • Share ›

sana • a year ago
how we make a spelling check program /code on c language.
• Reply • Share ›

admin • a year ago
Thanks for the post really helpful i have done it in another please have a look
http://ioeengineer.blogspot.co...
3 • Reply • Share ›

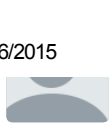
sana ➚ admin • a year ago
please reply fast
• Reply • Share ›

sana ➚ admin • a year ago
please do fast
• Reply • Share ›

johd • a year ago
Write a
program that reads a floating-point number and prints the ceiling, floor and rounded value. You should have three different functions that accept the inputs. In the main function, there should have three variables that accept the returned values from the respective functions. Test the program you created with the following data: 123.456789 123.499999 123.500001...

please help me code this one..I'm just a C language beginner :(
• Reply • Share ›

groomnestle • a year ago
how about below



now about below

```
int pow(int x, int n)
{
int sum = 1, i;
for(i=1;i<=n;i++)
sum*=x;


return sum;
}
```

which takes O(n).

5 ^ | v • Reply • Share ›

vb ➚ groomnestle • a year ago

Doesnt handle n with negative values

^ | v • Reply • Share ›

joeshmo • a year ago

Here's an iterative version of it:

//Time Complexity of optimized solution: O(logn)
#include<stdio.h>

float power(float x, int y)
{
float temp;
if(y == 0)
return 1;
temp = power(x, y/2);
if (y%2 == 0)
return temp*temp;
else
{
if(y > 0)
return x*temp*temp;
else

see more

^ | v • Reply • Share ›

Leo • a year ago

Dynamic

```
private static int power(int x, int y)

{

if (x < 0)

return 0;

int[] powers = new int[y + 1];

powers[0] = 1;

for (int i = 1; i<=y; i++)

{

powers[i] = powers[i-1]*x;

}

return powers[y];

}
```

• Reply • Share ›

Hitesh ➚ Leo • 8 months ago

Good one.

• Reply • Share ›

Leonidas Giannoulis • a year ago

```
uint64_t power(uint64_t b,uint8_t e)
{
return e?e%2?power(b*b,e/2)*b:power(b*b,e/2):1;
}
```

• Reply • Share ›

Tarun Kumar • a year ago

how to solve this if number overflow is there? n^k and k is large, say power(2, 40);

• Reply • Share ›

Hitesh ➚ Tarun Kumar • 8 months ago

Use some other data structure. In C, you can use long long. There should be some similar method in C as Java has BigDecimal/BigInteger, too.

• Reply • Share ›

rocky • 2 years ago

The complexity of the first program can be achieved as O(logn) by:
/*int power(int x,int n)
{
if(n==0)
return 1;
if(n==1)
return x;
else
if(n%2==0)
return power(x*x,n/2);
else
return power(x*x,n/2)*x;
}*/
Am i correct?

16 ^ | v • Reply • Share ›

ljk • 2 years ago

In case of negative y, what's wrong with just:

```
 y<0?printf("%f",1/pow(x,y*-1));:printf("%f",pow(x,y));
return 0;
```

4 ^ | v • Reply • Share ›

Raheel Shahzad • 2 years ago

{{{
double power(double x, double y).
{
double runner = x;.
double result = 1.0;.

if (y < 0).
{
return 1.0 / power(x, -y);.
}
int p = (int)y;.
y -= (double)p;.
while (p!= 0) {.
if (p & 1) {.
result *= runner;.
}
p>>=1;

runner *= runner;.

see more

• Reply • Share ›

AG • 2 years ago

The method suggested doesn't work for negative exponent.

• Reply • Share ›

DexterLtd • 3 years ago

No doubt the use of Temp variable reduces the complexity but still the constant factor is too large.
This code can be further improved by using a transcript array rather than a temporary variable.(Giving this problem a touch of dynamic programming solution)

The transcript array can be global or passed as argument.

• Reply • Share ›

DexterItd ↗ DexterLtd • 3 years ago

This code is same efficient as the temp variable one if we run it once.
for multiple call of pow() function, this will work fast cause of transcript.

• Reply • Share ›

KJD ↗ DexterItd • 2 years ago

The function is not generic enough to be called multiple times. Since x is fixed. If the function is called with different value of x and same value of y 2nd time it will give incorrect result.
example:
>>first call pow(2,1)returns 1
>>second call pow(5,1)returns 1 again (wrong).

1 • Reply • Share ›

KJD ↗ KJD • 2 years ago

correction in above comment both the times returned value will be 2 and not 1.

• Reply • Share ›

sana ↗ KJD • a year ago

can you help me a code

• Reply • Share ›

sana ➚ sana • a year ago

how we make a game on c language can you help me

• Reply • Share ›

DexterItd ➚ DexterLtd • 3 years ago

little correction in the code

```
int pow(int x,int y)
{
    if ( y == 0 )
       return 1;

    if ( Transcript[y] != 0)
        return Transcript[y];

    if ( y == 1)
    {
       Transcript[1] = x;
    }
    else if( y%2 == 0)
    {
       Transcript[y] = pow(x,y/2)*pow(x,y/2);
    }
    else
    {
       Transcript[y] = pow(x,y/2)*pow(x,y/2)*x;
    }

    return Transcript[y];
}
```

• Reply • Share ›

Sai Nikhil ➚ DexterItd • 2 years ago

this is again O(n) and no way better than above O(lg(n)) implementation.

• Reply • Share ›

brahma ➚ Sai Nikhil • 10 months ago

in the above code by using temp variable we can reduce complexity to O(logn) and little constant factor

• Reply • Share ›

- Interview Experiences
- Advanced Data Structures
- Dynamic Programming
- Greedy Algorithms
- Backtracking
- Pattern Searching
- Divide & Conquer
- Mathematical Algorithms
- Recursion
- Geometric Algorithms

Popular Posts

- All permutations of a given string
- Memory Layout of C Programs
- Understanding “extern” keyword in C
- Median of two sorted arrays
- Tree traversal without recursion and without stack!
- Structure Member Alignment, Padding and Data Packing
- Intersection point of two Linked Lists
- Lowest Common Ancestor in a BST.
- Check if a binary tree is BST or not
- Sorted Linked List to Balanced BST

- Follow @GeeksforGeeks

- # Recent Comments

  - Ashish Aggarwal

    Try Data Structures and Algorithms Made Easy -...

    Algorithms · 17 minutes ago

  - Vlad

    Thanks. Very interesting lectures.

    Expected Number of Trials until Success · 1 hour ago

  - cfh

    My implementation which prints the index of the...

    Longest Even Length Substring such that Sum of First and Second Half is same · 1 hour ago

  - Gaurav pruthi

    forgot to see that part ;)

    Bloomberg Interview | Set 1 (Phone Interview) · 1 hour ago

  - saeid aslami

    thanks

    Greedy Algorithms | Set 7 (Dijkstra's shortest path algorithm) · 1 hour ago

  - Cracker

    Implementation:...

    Implement Stack using Queues · 2 hours ago