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 you own Power without using multiplication(*) and division(/) operators

Method 1 (Using Nested Loops)

We can calculate power by using repeated addition.

For example to calculate 5^6.
1) First 5 times add 5, we get 25. (5^2)
2) Then 5 times add 25, we get 125. (5^3)
3) Then 5 time add 125, we get 625 (5^4)
4) Then 5 times add 625, we get 3125 (5^5)
5) Then 5 times add 3125, we get 15625 (5^6)

```
/* Works only if a >= 0 and b >= 0  */
int pow(int a, int b)
{
  if (b == 0)
    return 1;
  int answer = a;
  int increment = a;
  int i, j;
  for(i = 1; i < b; i++)
  {
     for(j = 1; j < a; j++)
     {
        answer += increment;
     }
     increment = answer;
  }
  return answer;
}
```

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

```
/* driver program to test above function */
int main()
{
  printf("\n %d", pow(5, 3));
  getchar();
  return 0;
}
```

Method 2 (Using Recursion)

Recursively add a to get the multiplication of two numbers. And recursively multiply to get *a* raise to the power *b*.

```
#include<stdio.h>
/* A recursive function to get a^b
  Works only if a >= 0 and b >= 0  */
int pow(int a, int b)
{
   if(b)
     return multiply(a, pow(a, b-1));
   else
    return 1;
}

/* A recursive function to get x*y */
int multiply(int x, int y)
{
   if(y)
     return (x + multiply(x, y-1));
   else
     return 0;
}

/* driver program to test above functions */
int main()
{
  printf("\n %d", pow(5, 3));
  getchar();
  return 0;
}
```

Please write comments if you find any bug in above code/algorithm, or find other ways to solve the same problem.

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

Related Tpoics:

- Backtracking | Set 8 (Solving Cryptarithmetic Puzzles)
- Tail Recursion
- Find if two rectangles overlap
- Analysis of Algorithm | Set 4 (Solving Recurrences)
- Print all possible paths from top left to bottom right of a mXn matrix
- Generate all unique partitions of an integer
- Russian Peasant Multiplication
- Closest Pair of Points | O(nlogn) Implementation

0 Tweet 0 0

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

11 Comments GeeksforGeeks

Sort by Newest ▾

Join the discussion

Join the discussion…

Anon • 18 days ago

Nothing is gained in this solution with recursion. Solution still has the same co stack each recursive step, resulting in higher memory usage.

This solution runs in O(a+b) time, but it's possible to get it in O(loga+logb) tim

http://pastebin.com/A1wbNdku

Anon • a month ago

can make reduce time complexity by using a divide and conquer approach like

Ankur Jain • 4 months ago

```
//
#include <cstdio>

#include <cmath>

#include <cstdlib>

int multiply(int x,int y)

{

int ans=0;

for (int i = 0; i < y; ++i)

{

ans+=x;
```

705

Subscribe

Recent Comments

Abhi You live US or India?

Google (Mountain View) interview · 17 minutes ago

Aman Hi, Why arent we checking for conditions...

Write a C program to Delete a Tree. · 57 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 1 hour ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...

Root to leaf path sum equal to a given number · 1 hour ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...

Count trailing zeroes in factorial of a number · 1 hour ago

newCoder3006 If the array contains negative numbers also. We...

Find subarray with given sum · 1 hour ago

```
ans *=x;

}
```

see more

1

abhishek08aug • a year ago

```
 #include<stdio.h>

int power(int a, int b) {
  if(a==0) {
    return a;
  }

  if(a==1) {
    return a;
  }

  if(b==0) {
    return 1;
  }

  if(b==1) {
    return a;
  }
```

see more

Balasubramanian • a year ago

The following snippet works for negative numbers as well. I tested it with a few

find any mistake,please comment.

```
 double myPow(int m,int n){
        if(n==0)
                return 1;
        bool isNeg=( n<0 )? true : false;
        n=abs(n);
        double ans=m;
        for(int i=2;i<=n;++i)
                ans=multiply(ans,m);
        if(isNeg)
                return 1/ans;
        return ans;
}

int multiply(int x,int y){
        if(y==0)
                return 0;
        if(y>0)
                return x+multiply(x,y-1);
        return -multiply(x,-y);
}
```

Thanks,

help • 3 years ago

what is the complexity of the recursion solution and how to calculate it?

anji.swe ➦ help • 2 years ago

I think for the first method time complexity is O(n^2) as it is using 2 for correct me if iam wrong !!

arjun1296 • 4 years ago

IMHO, Recursion should be avoided as far as possible. Recursion is just a be
design in computers make recursion costly. Time and Space for the creation
inefficient and unpredictable use of system stack. But still i love to see recursi

Nishant • 4 years ago

```
 #include<stdio.h>
int multi(int a,int b)
{
        int c=0;
        while(b)
        {
                if((b&1)!=0)
                c=c+a;
                a <>= 1;
        }
        return c;
}
int power(int a,int n)
{
        int i,j=1;
        for(i=1;i<=n;i++)
        j=multi(j,a);
        return j;
}
int main()
{
        printf("%d\n",power(2,10));
        return 0;
}
```

Sambasiva • 4 years ago

Recursion could be write like below

```
 int power(int a, int b)
{
    if(b == 0)
        return 1;

    int inc = pow(a, b - 1);

    int ret = 0;

    int i = 0;
    for(; i < a; ++i)
        ret += inc;

    return ret;
}
```

Ahmet Alp Balkan • 4 years ago

Recursion is the better and elegant solution imho. Is there a way to handle <0 wondering.