Compute the minimum or maximum of two integers without branching

On some rare machines where branching is expensive, the below obvious approach to find minimum can be slow as it uses branching.

```
/* The obvious approach to find minimum (involves branching) */
int min(int x, int y)
{
  return (x < y) ? x : y
}
```

Below are the methods to get minimum(or maximum) without using branching. Typically, the obvious approach is best, though.

Method 1(Use XOR and comparison operator)

Minimum of x and y will be

```
y ^ ((x ^ y) & -(x < y))
```

It works because if x < y, then -(x < y) will be all ones, so r = y ^ (x ^ y) & ~0 = y ^ x ^ y = x. Otherwise, if x >= y, then -(x < y) will be all zeros, so r = y ^ ((x ^ y) & 0) = y. On some machines, evaluating (x < y) as 0 or 1 requires a branch instruction, so there may be no advantage.

To find the maximum, use

```
x ^ ((x ^ y) & -(x < y));
```

```
#include<stdio.h>
```


```
/*Function to find minimum of x and y*/
int min(int x, int y)
{
  return y ^ ((x ^ y) & -(x < y));
}

/*Function to find maximum of x and y*/
int max(int x, int y)
{
  return x ^ ((x ^ y) & -(x < y));
}

/* Driver program to test above functions */
int main()
{
  int x = 15;
  int y = 6;
  printf("Minimum of %d and %d is ", x, y);
  printf("%d", min(x, y));
  printf("\nMaximum of %d and %d is ", x, y);
  printf("%d", max(x, y));
  getchar();
}
```

Method 2(Use subtraction and shift)
If we know that

```
INT_MIN <= (x - y) <= INT_MAX
```

, then we can use the following, which are faster because (x - y) only needs to be evaluated once.

Minimum of x and y will be

```
y + ((x - y) & ((x - y) >>(sizeof(int) * CHAR_BIT - 1)))
```

This method shifts the subtraction of x and y by 31 (if size of integer is 32). If (x-y) is smaller than 0, then (x -y)>>31 will be 1. If (x-y) is greater than or equal to 0, then (x -y)>>31 will be 0.
So if x >= y, we get minimum as y + (x-y)&0 which is y.
If x < y, we get minimum as y + (x-y)&1 which is x.

Similarly, to find the maximum use

```
x - ((x - y) & ((x - y) >> (sizeof(int) * CHAR_BIT - 1)))
```


Popular Posts

```
x - ((x - y) & ((x - y) >> (sizeof(int)  CHAR_BIT - 1)))
```

```
#include<stdio.h>
#define CHAR_BIT 8

/*Function to find minimum of x and y*/
int min(int x, int y)
{
  return  y + ((x - y) & ((x - y) >>
            (sizeof(int) * CHAR_BIT - 1)));
}

/*Function to find maximum of x and y*/
int max(int x, int y)
{
  return x - ((x - y) & ((x - y) >>
            (sizeof(int) * CHAR_BIT - 1)));
}

/* Driver program to test above functions */
int main()
{
  int x = 15;
  int y = 6;
  printf("Minimum of %d and %d is ", x, y);
  printf("%d", min(x, y));
  printf("\nMaximum of %d and %d is ", x, y);
  printf("%d", max(x, y));
  getchar();
}
```

Note that the 1989 ANSI C specification doesn't specify the result of signed right-shift, so above method is not portable. If exceptions are thrown on overflows, then the values of x and y should be unsigned or cast to unsigned for the subtractions to avoid unnecessarily throwing an exception, however the right-shift needs a signed operand to produce all one bits when negative, so cast to signed there.

Source:
http://graphics.stanford.edu/~seander/bithacks.html#IntegerMinOrMax

Related Tpoics:

- Check if a number is multiple of 9 using bitwise operators
- How to swap two numbers without using a temporary variable?
- Divide and Conquer | Set 4 (Karatsuba algorithm for fast multiplication)
- Find position of the only set bit
- Swap all odd and even bits
- Add two bit strings
- Write your own strcmp that ignores cases
- Binary representation of a given number

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

34 Comments GeeksforGeeks

Recent Comments

Abhi You live US or India?

Google (Mountain View) interview · 22 minutes ago

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

Write a C program to Delete a Tree. · 1 hour 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

Sort by Newest

Join the discussion…

Rohit • 3 months ago

correct me if i'm wrong:
but does y + (x-y)&1 give you back y ?
This is supposed to be bit wise...& operator and 1 is like 0x00000001 right?

guest • 4 months ago

```
int Max(int a, int b)
{
int c=0;
while(x || y)
{
c++;
x--;
y--;
}
return c;
}

int min(int a, int b)
{
int c =0;
while(x&&y) { c++; x--; y--; }
return c;
}
```

Is this allowed ??

Chao • 4 months ago

I don't agree with one sentence "If (x-y) is smaller than 0, then (x -y)>>31 will b
is one, then it should be -1). There are lots of the same errors in this website.

ankit sahu • 7 months ago

awesome...second logic...

Jekin • 7 months ago

Error in method 2 :

```
y + ((x - y) & ((x - y) >>(sizeof(int) * CHAR_BIT - 1)))
```

There must be && (logical) instead of &(bitwise)

rahulrk1991 • 9 months ago

1.Find the mean of the 2 numbers.(say x)
2.Find the absolute value of the difference between the 2 numbers(say y).(abs
branching,see bottom note).
3.Smaller of the 2 numbers will be x-(y/2).
4.Larger of the 2 numbers will be x+(y/2).

Note:Absolute value of a number n is squareroot(n*n).

1 

François Fauster El Be • 10 months ago

Rohit Verma it is right but its potentially slower than a simple ternary condition

François Fauster El Be • 10 months ago

Noway Palmero

gargsanjay • 10 months ago

we can use
y + ((x - y) & -((y-x) >> (sizeof(int) * CHAR_BIT - 1)))
for general c compiler
whatsay??

gargsanjay • 10 months ago

x&1 is x or 1??

```
/* Paste your code here (You may delete these lines if not writing c
```

NNavneet • a year ago

I think we can use this method also ,this is much more easy to understand . P

```
#include<stdio.h>
#include<iostream>
#define CHAR_BIT 8

using namespace std;

int main()
{
```

```
  int y = 21;



  int d= x-y;
  int t = ((x-y)>>31)&1;
  cout<<t<<"   "<<d<<endl;
  cout<<"maximum number is :"<<x-d*t<<endl;
  cout<<"minimum number is :"<<y+d*t<<endl;

  getchar();
}
```

ANONYMOUS • a year ago

If x < y, we get minimum as y + (x-y)&1 which is x.

y+(x-y)&1 is (y+1)...how is it x?

[sourcecode language="C"]
/* Paste your code here (You may delete these lines if not writing code) */

Jayant Mukherji • a year ago

NO .. if(x==y) max(x,y) will return 0

Animesh Mishra • a year ago

Ankit Singhal : What if I told you.... writing 7 &#039w&#039 in slow doesn&#03
tum rehne do Tumse na ho payega ;-)

Vaibhav Mishra • a year ago

arre thelu tumse naa ho payega... :P

Milan Pandey • a year ago

KAM CODE KAR BHAI, KAM CODE KAR!!

Ankit Agarwal • a year ago

hii thelu u r pro!!

Ramasubramani • a year ago

int max = (x>y)*x + (y>x)*y;
int min = (x>y)*y + (y>x)*x;

Shyam • 2 years ago

@geeksforgeeks Can you explain the XOR method with examples?

```
/* Paste your code here (You may delete these lines if not writing c
```

Neha • 2 years ago

```
#include <stdio.h>

unsigned int leftmostBit(int n){
      int aux;
      while(n){
      aux = n;
```

```
      n &= n-1;
   }
   return aux;
}

int MaxMin(int a, int b){
   int xxor = a ^ b;
   int aux = leftmostBit(xxor);
   return ((a&aux)*a + (b&aux)*b)/aux;
}

int main(){
   int a=10, b=6;
   printf("\n%d ", MaxMin(a, b));
   return 0;
}
```

Uday • 2 years ago

Can anyone explain " when x<y and it&#039s true then it should return value 1
But Above it is mentioned that -(x<y) will return all Ones.

GeeksforGeeks ➚ Uday • 2 years ago

@Uday: When x is smaller than y, the value of expression -(x < y) bec
(integer number) contains all 1s.

Cristobal • 2 years ago

hello

very interesting article
have you tried this on GPUs, any speedup improvement for conditionals?

```
/* Paste your code here (You may delete these lines if not writing co
```

sureshpaldia22 • 3 years ago

It can also be done using module operator..

```
int compare(int num1,int num2)
{
    return (num1 % num2) == num1 ? num2 : num1;
}
```

Venki ➦ sureshpaldia22 • 3 years ago

@Suresh, you are using ternary operator, which finally ends in branchi

Vinay Kumar • 4 years ago

mask = (a-b)>>31
Min = mask & a| ~mask & b
Max = mask & b| ~mask & a

Saravanan Mani • 4 years ago

```
int max(int a, int b)
{
   return (!!(b/a))*b+(!!(a/b))*a;
}
```

coderyogi • 4 years ago

In method 1, the function names for max and min do the opposite, i.e., find the

vindhya coderyogi • 2 years ago

@coderyogi
yes..u r ryt...the functions give the opposite results...it should be
printf("min= %d",y^((x^y)&(x<y)));
printf("max= %d",x^((x^y)&(x<y)));

[sourcecode language="C"]
/* Paste your code here (You may delete these lines if not writing code

Shekhu coderyogi • 4 years ago

The methods look fine to me. I mean they do what their name suggests

ar • 4 years ago

int max(int a, int b){
return (a>=b)*a + (b>a)*b;
}

GeeksforGeeks • 4 years ago

@a2ms: Branching refers to the logic where we have multiple paths for the ne decides which path to follow.

For example, if else, ternary operator : ?, etc..

a2ms • 4 years ago

what is branching?

Jekin a2ms • 7 months ago

Conditional operators. Ex. if, switch, ?:

Subscribe Add Disqus to your site