What's wrong with this Pollard Rho implementation

```
#include <iostream>
#include <cstdlib>
typedef  unsigned long long int ULL;
ULL gcd(ULL a, ULL b)
{
   for(; b >0 ;)
   {
       ULL rem = a % b;
       a = b;
       b = rem;
   }
   return a;
}
void pollard_rho(ULL n)
{
   ULL i = 0,y,k,d;
   ULL *x = new ULL[2*n];
   x[0] = rand() % n;
   y = x[0];
   k = 2;
   while(1){
       i = i+1;
       std::cout << x[i-1];
       x[i] = (x[i-1]*x[i-1]-1)%n;
       d = gcd(abs(y - x[i]),n);
       if(d!= 1 && d!=n)
          std::cout <<d<<std::endl;
       if(i+1==k){
         y = x[i];
         k = 2*k;
       }
   }
}

int main()
{
```

This implementation is derived from CLRS 2nd edition (Page number 894). `while(1)` looks suspicious to me. What should be the termination condition for the while loop?

I tried `k<=n` but that doesn't seem to work. I get segmentation fault. What is the flaw in the code and how to correct it?

c++ | algorithm | factorization | clrs

asked May 18 '11 at 5:52

Pointer

6 1

You are right about the `while` loop; there is no termination so `i` becomes large and indexes into `x` with an illegal value which causes the segmentation fault. – Richard Schneider May 18 '11 at 5:57

So what should be the terminating condition? Please check out Pollard Rho's algorithm in CLRS. – Pointer May 18 '11 at 5:59

Is this a trick question? How should we know what's wrong with it? – Gabe May 18 '11 at 5:59

No its not a trick question. I want to know how to correct the code. – Pointer May 18 '11 at 6:01

add comment

3 Answers

Try replacing `while(1) { i = i + 1;` with this:

```
for (i = 1; i < 2*n; ++i) {
```

answered May 18 '11 at 6:01

Richard Schneider
14.6k 1 19 36

doesn't seem to work. – Pointer May 18 '11 at 6:08

add comment

I only have a 1st edition of CLRS, but assuming it's not too different from the 2nd ed., the answer to the termination condition is on the next page:

> This procedure for finding a factor may seem somewhat mysterious at first. Note, however, that POLLARD-RHO never prints an incorrect answer; any number it prints is a nontrivial divisor of *n*. POLLARD-RHO may not print anything at all, though; there is no guarantee that it will produce any results. We shall see, however, that there is good reason to expect POLLARD-RHO to print a factor of *p* of *n* after approximately sqrt(*p*) iterations of the **while** loop. Thus, if *n* is composite, we can expect this procedure to discover enough divisors to factor *n* completely after approximately $n^{1/4}$ update, since every prime factor *p* of *n* except possibly the largest one is less than sqrt(*n*).

So, technically speaking, the presentation in CLRS doesn't have a termination condition (that's probably why they call it a "heuristic" and "procedure" rather than an "algorithm") and there are no guarantees that it will ever actually produce anything useful. In practice, you'd likely want to put some iteration bound based on the expected $n^{1/4}$ updates.

answered May 18 '11 at 14:47

mhum
1,580 6 9

add comment

Why store all those intermediary values? You really don't need to put x and y in a array. Just use 2 variables which you keep reusing, `x` and `y`.

Also, replace `while(1)` with `while(d == 1)` and cut the loop before

```
if(d!= 1 && d!=n)
     std::cout <<d<<std::endl;
  if(i+1==k){
    y = x[i];
    k = 2*k;
```

So your loop should become

```
while(d == 1)
{
    x = (x*x - 1) % n;
    y = (y*y - 1) % n;
    y = (y*y - 1) % n;
    d = abs(gcd(y-x,n))%n;
}
if(d!=n)
  std::cout <<d<<std::endl;
else
  std::cout<<"Can't find result with this function \n";
```

Extra points if you pass the function used inside the loop as a parameter to `pollard`, so that if it can't find the result with one function, it tries another.

edited Oct 10 '13 at 18:08

answered May 18 '11 at 6:32

Cronco
431 2 10

1 Hm. The `while` terminates when `d != 1`, so isn't the `d != 1` in the `if` statement redundant? – estan Oct 10 '13 at 17:16

@estan good find, will modify :) – Cronco Oct 10 '13 at 18:08

add comment

Not the answer you're looking for? Browse other questions tagged c++ algorithm factorization clrs **or ask your own question.**