Category: math.NT

Three Triangles Integer Factorization Algorithm

This is my discovery of an algorithm for integer factorization. To my knowledge it is not based on any existing solutions. I do not claim it to be efficient or useful, I'm only concerned with its correctness and completeness.

Abstract

Factor the composite number $C=(a+1)*(a+b)$ by finding the solution to $C=T(a)+T(a+b)-T(b-1)$ where $T(n)$ generates the nth triangular number.

Definitions

- **a, b := natural numbers**
- **T(n) := n*(n+1)/2**
- **S(a, b) := T(a)+T(a+b)-T(b-1)**
- **make_even(n) := if n is odd return n-1 else return n**

Proof of $(a+1)*(a+b)=T(a)+T(a+b)-T(b-1)$

1. $(a+1)*(a+b)=T(a)+T(a+b)-T(b-1)$
2. $a*a+a*b+a+b=a*(a+1)/2+(a+b)*(a+b+1)/2-(b-1)*b/2$
3. $2*a*a+2*a*b+2*a+2*b=a*(a+1)+(a+b)*(a+b+1)-(b-1)*b$
4. $2*a*a+2*a*b+2*a+2*b=a*a+a+a*(a+b+1)+b*(a+b+1)-b*b+b$
5. $2*a*a+2*a*b+2*a+2*b=a*a+a+a*a+a*b+a+b*a+b*b+b-b*b+b$
6. $2*a*a+2*a*b+2*a+2*b=2*a*a+2*a*b+2*a+2*b$
7. $0=0$

Proof that $S(a, b) > S(a-2, b+2)$

This is required to show that the algorithm makes progress on step 3.

1. $S(a, b) > S(a-2, b+2)$
2. $(a+1)*(a+b) > (a+1-2)*(a+b-2+2)$
3. $a*a+a+b > (a-1)*(a+b)$
4. $a*a+a+b > a*a-(a+b)$
5. $a+b > -(a+b)$
6. $1*(a+b) > -1*(a+b)$
7. **Since a, b > 0 =>** $1 > -1$

Proof that $S(a, b) < S(a, b+max(1, ceil((C-S(a, b))/(a+1))))$

This is required to show that the algorithm makes progress on step 4.

1. $S(a, b) < S(a, b+max(1, ceil((C-S(a, b))/(a+1))))$
2. $(a+1)*(a+b) < (a+1)*(a+b+max(1, ceil((C-S(a, b))/(a+1))))$
3. $a+b < a+b+max(1, ceil((C-S(a, b))/(a+1)))$
4. $0 < max(1, ceil((C - S(a, b))/(a+1)))$
5. $0 < 1$

Algorithm

input: **C => integer greater than 2**
output: **found factors, or 1 and C if C is a prime number**

1. **let a = make_even(floor(sqrt(C)) - 1)**
2. **let b = 1**
3. **if** $S(a, b) > C$ **then** $a = a - 2, b = b + 2$
4. **if** $S(a, b) < C$ **then** $b = b + max(1, ceil((C - S(a, b)) / (a + 1)))$
5. **if** $S(a, b) == C$ **then exit: found factors** $(a + 1)$ **and** $(a + b)$
6. **if** $a == 0$ **then exit:** C **is a prime number**
7. **goto step 3.**

Examples

Factor 51

1. **a = 6, b = 1**
   S(6, 1) = 49
   => b = b + max(1, ceil(2 / 7))
2. **a = 6, b = 2**
   S(6, 2) = 56
   => a = a - 2, b = b + 2
3. **a = 4, b = 4**
   S(4, 4) = 40
   => b = b + max(1, ceil(11 / 5))
4. **a = 4, b = 7**
   S(4, 7) = 55
   => a = a - 2, b = b + 2
5. **a = 2, b = 9**
   S(2, 9) = 33
   => b = b + max(1, ceil(18 / 3))
6. **a = 2, b = 15**
   S(2, 15) = 51
   => 51 = 3 * 17

Factor 23

1. **a = 2, b = 1**
   S(2, 1) = 9
   => b = b + max(1, ceil(14 / 3))
2. **a = 2, b = 6**
   S(2, 6) = 24
   => a = a - 2, b = b + 2
3. **a = 0, b = 8**
   => 23 = 1 * 23

Conclusion

Instead of focusing on performance of factorization this algorithm tries to explore a new solution. One potential benefit I see is the reduced magnitude of dividends used in divisions. Hope it inspires some ideas.