Problem 4

Best post cache update: 60 minutes

Showing posts 1 to 25 out of 25

5/5 Kudos remaining

Mon, 23 May 2005, 21:26
Begoner
Python

Quote Report 564

```
You can also do this with pen and paper.  You have a number:

(100a + 10b + c)(100d + 10e + f)

Which is a palindrone.  This factors out to:

                   100cd + 10ce + cf +
          1000bd + 100be + 10bf +
10000ad + 1000ae + 100af

Assuming the first digit is 9, then cf must be equal to nine.
Also, both a and d must then be equal to nine.  The only ways
to make the last digit of the product of two integers 9 would
be if those integers were either of:

1 and 9
3 and 3
7 and 7

So, both numbers must start with 9, end with either 1, 3, 7,
or 9, and one of them must be divisible by 11.  The only
numbers divisible by 11 from 900 - 1000 are:

902
913
924
935
946
957
968
979
990

You can discard all of those that do not end in either 1, 3,
7, or 9, and you are left with:

913
957
979

So now the presumed answer is either:

(900 + 10 + 3)(900 + 10x + 3)
```

(900 + 50 + 7)(900 + 10x + 7)
(900 + 70 + 9)(900 + 10x + 1)

Factoring all those out, you get:

810000 + 9000x + 2700 + 9000 + 100x + 30 + 2700 + 30x + 9
824439 + 9130x

Now, for the first digit 824439 + 9130x to be 9, x must be 9 (if x were 8, then 824439 + 9130x = 897479, and the first digit is 8). And so you have 913 * 993, which is the answer. You can factor the others out to see if they produce a bigger answer, which they don't.

✓

Mon, 23 May 2005, 00:15
Begoner
Python

Quote Report 329

The palindrome can be written as:

abccba

Which then simpifies to:

100000a + 10000b + 1000c + 100c + 10b + a

And then:

100001a + 10010b + 1100c

Factoring out 11, you get:

11(9091a + 910b + 100c)

So the palindrome must be divisible by 11. Seeing as 11 is prime, at least one of the numbers must be divisible by 11. So brute force in Python, only with less numbers to be checked:

Python **Hide Code**

```
def c():
        max = maxI = maxJ = 0
        i = 999
        j = 990
        while (i > 100):
                j = 990
                while (j > 100):
                        product = i * j
                        if (product > max):
                                productString = str(product)
                                if (productString == productString[::-1]):
                                        max = product
                                        maxI = i
                                        maxJ = j
                j -= 11
            i -= 1
        return max, maxI, maxJ
```

Returns an answer in 0.016 secs.

✓

Sun, 22 Aug 2004, 04:48
etatsui
C#

Quote Report 159

```
The palindrome can be written as
11(9091a + 910b + 100c) = mn;
a,b & c being 1 digit integers and m & n being 3 digit
intergers.

Let 11 * 10 < m < 11 * 90;

for(int a=9; a>=1; a--)
  for(int b=9; b>=0; b--)
    for(int c=9; c>=0; c--){
      num = 9091 * a + 910 * b + 100 * c;
      for(int divider=90; divider>=10; divider--){
        //look for divider that can divide
        //and also doesn't make n > 999
        if((num % divider) == 0){
          if((num / divider) > 999)
            break;
          else
            result = num * 11; //Found it!
        } else { break; }
      }
```

✓

Tue, 17 Jan 2006, 13:54
iang
Python

Quote Report 80

Python one-liner

Python **Hide Code**

```
max([x*y for x in range(900,1000) for y in range(900,1000) if str(x*y) ==
str(x*y)[::-1]])
```

✓

Fri, 31 Dec 2004, 17:03
REBoss
APL/J/K

Quote Report 64

```
In J:
>([:{: ]#~ (=|.&.>)) <@":"0 /:~(0:-.~[:,>:/**/)~(i.100)-.~i.1000
```

✓

Sun, 25 Sep 2011,

10:35
Lucy_Hedgehog
Python

Quote Report 41

I have a few more solutions for different sizes:

```
# n=2  9009 = 91 * 99
# n=3  906609 = 913 * 993
# n=4  99000099 = 9901 * 9999
# n=5  9966006699 = 99681 * 99979
# n=6  999000000999 = 999001 * 999999
# n=7  99956644665999 = 9997647 * 9998017
# n=8  9999000000009999 = 99990001 * 99999999
# n=9  999900665566009999 = 999980347 * 999920317
# n=10 99999834000043899999 = 9999996699 * 9999986701
# n=11 9999994020000204999999 = 99999996349 * 99999943851
# n=12 999999000000000000999999 = 999999999999 * 999999000001
# n=13 99999963342000024336999999 = 9999999993349 * 9999996340851
# n=14 9999999000000000000009999999 = 99999999999999 *
99999990000001
# n=15 999999974180040040081479999999 = 999999998341069 *
999999975838971
# n=16 99999999000000000000000099999999 = 9999999999999999 *
9999999900000001
# n=18 999999999470552640046255074999999999 = 999999999889625119
* 999999999580927521
# n=19 99999999988837057200275073888999999999 =
9999999999632783059 * 9999999999250922661
# n=20 9999999999694448232002328444969999999999 =
99999999998547088359 * 99999999998397393961
```

The following code find the largest palindrom for all n <= 16 in less than 1 minute.

Python **Hide Code**

```python
# Using python 3.0

def ispalindrome(x):
    s = str(x)
    return s == s[::-1]


def inverse(x,mod):
    """Compute the modular inverse of x modulo a power of 10.
    Return None if the inverse does not exist.
    This function uses Hensel lifting."""
    a = [None, 1, None, 7, None, None, None, 3, None, 9][x%10]
    if a == None: return a
    while True:
        ax = a*x % mod
        if ax == 1: return a
        a = (a * (2 - ax)) % mod

def pal2(n):
    assert n > 2

    # Get a lower bound:
```

```
    # If n is even then we can construct a first palindrome.
    # If n is odd we simply guess
    k = n//2
    while True:
      maxf = 10**n - 1
      maxf11 = (maxf - 11) // 22 * 22 + 11
      minf = 10**n - 10**(n-k) + 1
      if 2*k == n:
        best = maxf * minf
        factors = (maxf, minf)
        assert ispalindrome(best)
      else:
        best = minf * minf
        factors = None
      # This palindrome starts with k 9's.
      # Hence the largest palindrom must also start with k 9's and
      # therefore end with k 9's.
      # Thus, if p = x * y is the solution then
      # x * y + 1 is divisible by mod.
      mod = 10**k
      for x in range(maxf11, 1, -22):
        if x * maxf < best:
            break
        ry = inverse(x, mod)
        if ry == None:
            continue
        maxy = maxf + 1 - ry
        for p in range(maxy * x, best, -x * mod):
            if ispalindrome(p):
                if p > best:
                    best = p
                    y = p//x
                    factors = (x, y)
      if factors:
        return best, factors
      else: k-=1
```

Wed, 5 Jan 2005, 08:26
bitRAKE
Assembly

Quote Report 40

I brute force of course:

Assembly **Show Code**

```
; maximum palidrome
        xor ebx, ebx

        ; two three digit numbers
        mov     esi, 999
_0:     mov     edi, 999
_1:     ; multiplied together
        mov ecx, esi
        imul ecx, edi

        ; Is palindrome?
        push ecx
        push ecx
        push ecx
        fild DWORD PTR [esp]
        fbstp TBYTE PTR [esp]

        ; five or six digits
        mov edx, DWORD PTR [esp]
        mov eax, edx
        and edx,  0F0F0Fh
        and eax, 0F0F0F0h
        cmp DWORD PTR [esp], 100000h
        jc five
; six digits
        ; 00 AB CD EF
        shl edx, 4+8
        shl eax, 4
        or eax, edx
        bswap eax
        ; 00 FE DC BA
        jmp check

five:   ; 00 0A BC DE
        shl edx, 8
        shl eax, 16
        or eax, edx
        bswap eax
        ; 00 0E DC BA
check:  cmp DWORD PTR [esp], eax
        lea esp, [esp + 12]
        jne @F
        cmp ebx, eax
        cmovc ebx, eax
@@:     dec edi
        cmp edi, 99
        jne _1
        dec esi
        cmp esi, 99
        jne _0
```

Sat, 17 Sep 2005, 04:14
Olathe
Haskell

Quote Report 35

In Ruby :

Haskell **Show Code**

```
max = 0
100.upto(999) { |a|
  a.upto(999) { |b|
    prod = a * b
    max = [max, prod].max if prod.to_s == prod.to_s.reverse
  }
}
puts "Maximum palindrome is #{ max }."
```

Thu, 3 Mar 2005, 18:09

Quote Report 29

In Java..

```
                    int max=0;
                    for(int i=100;i<=999;i++)
                            for(int j=100;j<=999;j++)
                                    if (palin(j*i))
```

```
                                        if(j*i>max)
                                                max=j*i;

                System.out.println(max);



Note: I've created 2 utility methods for Palindrome and Reverse

        public static long rev(long n)
        { // This method simply returns a reversed number

                String s=""+n;
                StringBuffer sb=new StringBuffer(s);
                sb=sb.reverse();
                s=""+sb;

                return Long.parseLong(s);
        }

        public static boolean palin(int n)
        {  //This method checks if a number is palin or no

                String s1=""+n;
                String s2=""+rev(n);
                if(s1.equals(s2))
                        return true;

                return false;
        }
```

✓

Thu, 10 Nov 2005, 20:10
quangntenemy
Java

Quote Report 28

Haskell

Haskell **Show Code**

```
[m | a <- [9], b <- [0..9], c <- [0..9], m <- [100001* a + 10010 * b + 1100 *
c], [x | x <- [100..999], m `mod` x == 0 && m `div` x < 1000] /= []]
```

✓

Wed, 24 Aug 2005, 16:30
ebgreen
PowerShell

Quote Report 18

My brute force Python solution. I am using these Math Challenges to learn Python, so my solutions are rarely the best. :)

Hide Code

```
def IsPalindrome(n):
        myStr = str(n)
        if myStr == myStr[::-1]:
                return 1
        else:
                return 0
import time

tStart = time.time()
nResult = 0
for i in range(100,1000):
    for j in range(100,1000):
        if IsPalindrome(i*j) == 1 and (i*j) > nResult:
            nResult = (i*j)
print "Winner = " + str(nResult)
print "run time = " + str((time.time() - tStart))
```

✓

Wed, 2 Feb 2005, 16:01
alereborn

Quote Report 15

brute force in python...

Hide Code

```
def ispalindrome(string):
        decide=1
        i=0
        while i<=int(len(string)/2) and decide==1:
                if string[i]!=string[-(i+1)]:
                        decide=0
                i+=1
        return decide

if __name__=='__main__':
    aux=0
    for k in range(101,1000):
        for j in range(101,1000):
                if ispalindrome(str(j*k)) and j*k>aux:
                        aux=j*k
    print aux
```

I would really appreciate if VRAbi and R.E.Boss could tell me what langauge are they using... and why is it so criptic...

✓

Sat, 17 Mar 2007, 00:10
yeus
C/C++

Quote Report 12

my highly optimized C++ -version which takes only 373 nanoseconds (=3.73*10^-7s on a 1.8GHz, optimized build, using microsofts C++ compiler (2005 version) :):

C/C++ **Show Code**

```
static inline unsigned int FindLargestPalindromicNumber()
{
        for(int x=9;x>=0;--x){  int a=x*100001;
        for(int y=9;y>=0;--y){  int b=a+y*10010;
        for(int z=9;z>=0;--z){  int n=b+z*1100;
                for(int i=990;i>99;i-=11){
                        if(n%i==0){
                                int t=n/i;
                                if(t<1000)return n;
        }}}}}
        return 0;
}
```

✓

Sun, 18 Sep 2005, 04:57
gel
C/C++

Quote Report 10

C - brute force

C/C++ **Show Code**

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

int main(void)
{
    int x;
    int y;
    int z;
    int max = 0;
    char a[7];
    char b[7];

    for(x = 999 ; x > 99 ; x--)
    {
        for(y = 999 ; y > 99 ; y--)
        {
            z = x * y;
            sprintf(a, "%d", z);
            strcpy(b, a);
            _strrev(b);
            if(strcmp(a, b) == 0)
            {
                if(z > max)
                {
                    max = z;
                }
            }
        }
    }
    printf("Answer = %d\n", max);
    return 0;
}
```

Very similar to Neitsa's.

✓

Tue, 9 May 2006, 21:44
andylaw
Perl

Quote Report 9

A slightly refined version in Ruby with a bit of applied thought. If we start from 999 and work DOWN (rather than the way that everyone uses by default of starting at 100 and working up) then we can home in on the solution quicker.

Also, by breaking out of the inner loop when we find a palindrome (because we aren't going to find a bigger value in this loop if we're decrementing the inner loop) we can save further time.

Finally, by stopping the search when the outer loop is less than the sqrt of the highest palindrome found thus far we can save ourselves even more time.

This cuts down the search space by more than 92% by my reckoning.

```
<div style='padding:.5em;border:1px solid black;background:#ffc;'
class='info'>
[tt=]
#!/usr/bin/ruby

# Euler problem 004

# Variables for recording the highest value seen
max = 0

# Iterate DOWN - because we'll find the highest value
# quicker that way
999.downto(100) do |n|

# If we already found something bigger than the square of our
# outer (higher) loop number then we can stop
        if (n*n < max) then
                break
        end
# Again, iterate DOWN
n.downto(100) do |m|

# calculate the product and check if it is a palindrome
# If it is, and it's bigger than the best one we've seen to now
# then record it as the biggest and break out of the inner loop
# because we aren't going to find a bigger value for this value
# in the outer loop
                i = (n * m)
                s = i.to_s
                if (s == s.reverse) then
                        if (i > max) then
                                max = i
                        end
                        break
                end
        end
end

# Once we're done, report the highest value seen
puts max
[/tt]</div>
```

✓

Thu, 5 Jan 2006, 10:18
paulj
Python

Quote Report 7

```
Python:

maxp = 0
for i in xrange(100,1000):
   for j in xrange(i, 1000):
      p = i * j
      if str(p) == str(p)[::-1] and p > maxp:
          maxp = p

print maxp
```

range for i was chosen to be 'all 3-digit numbers'. range for j is such that we don't end up with duplicates.

✓

Sun, 18 Sep 2005, 14:57
spuno
C#

Quote Report 7

C# solution

```
int sum;
int Largest = -1;
string res;
bool Palindrome = true;

for(int x=999;x>=1;x--)
{
        for(int y=999;y>=1;y--)
        {
                sum = x*y;
                res = sum.ToString();
                Palindrome = true;
                for(int t = 0;t<res.Length / 2;t++)
                {
                        if(res[t] != res[res.Length-t-1])
Palindrome = false;
                }
                if(Palindrome) if(sum > Largest) Largest = sum;
        }
}
Console.WriteLine(Largest);
Console.ReadLine();
```

✓

Mon, 24 Jul 2006, 16:12
thattommyhall
Clojure

Quote Report 6

paulj said

```
Python:

maxp = 0
for i in xrange(100,1000):
   for j in xrange(i, 1000):
      p = i * j
      if str(p) == str(p)[::-1] and p > maxp:
          maxp = p

print maxp
```

range for i was chosen to be 'all 3-digit numbers'. range for j is such that we don't end up with duplicates.

Like mine, just a shade more thoughtfull, beautiful.

✓

Mon, 26 Dec 2005, 19:07
VrAbi
APL/J/K

Quote Report 6

In K(kx.com):

Show Code

Refactored, to match olegyk's J solution

Show Code

✓

Sun, 16 Oct 2005, 16:36
elt
Haskell

Quote Report 6

Haskell solution

```
palindrome = maximum (filter (isPalindrome) [a*b | a <-
[100..999], b <- [a..999]])
        where
                isPalindrome a = a == merge (makeList a)
                        where
```

```
                                merge [] = 0
                                merge (x:xs) = x + merge (map (\x
-> x*10) xs)

                                makeList 0 = []
                                makeList a = (makeList (div a 10))
++ [(mod a 10)]
```

✓

Mon, 17 Oct 2005, 00:34
Silverfish
Python

Quote Report 6

My method is sort of brute force.

I started by creating a procedure (this is all in Maple), called palintest, that tests whether a given 6 digit number is palindrome:

Python **Hide Code**

```
palintest := proc(r) local a,b,c,tests;
b := []; c:= r;
for a from 1 to 6 do
b := [op(b),c mod 10]; c := floor(c/10);
end do;
for a from 1 to 3 do if b[a] <> b[7-a] then return (false); end if; end do;
return(true);
end proc;
```

Then, I set up a program to run through all reasonable pairs of 3 digit numbers, multiply them, and check if the product is a palindrome:

Python **Hide Code**

```
palinfind := proc() local a,b, x,y; x := [];
for a from 999 to 101 by -1 do print(a);
for b from min(floor(999999/a),999) to
max(ceil(100001/a),101) by -1 do
if palintest(a*b) then x := {a,b}; return(x); end if;
end do; end do;
return(x);
end proc;
```

By reasonable, I mean that the numbers must be 3 digit, so between 100 and 999, not including 100, as a number with 100 as a factor would end in 0, and so if it was a palindrome it would start with 0, which is impossible. Also, the product must be 6 digit, so between 100001 and 999999, the lowest and highest 6 digit palindromes.

This produces 583*995 = 580085. Given I started with the high numbers first, any higher palindromes would have one number less than 995 (as those higher have been covered, and those not covered for 995 would be those with the other number less than 583), and also both numbers must be at least 583, as if one is less, then the other would need to be greater than 995 (which is impossible). So we can alter the program a bit to find another palindrome, with this program:

Python **Hide Code**

```
palinfind2 := proc() local a,b, x,y; x := [];
for a from 999 to 583 by -1 do print(a);
for b from min(floor(999999/a),999) to
max(ceil(580086/a),583) by -1 do
if palintest(a*b) then x := {a,b}; return(x); end if;
end do; end do;
return(x);
end proc;
```

This gives 913*993 = 906609, and by a similar process we can look for any higher palindromes, which do not exist.

Sun, 25 Jul 2004, 18:49
rayfil
Assembly

Quote Report 6

Assembly **Show Code**

```
.data
      XXX         dd    100
      YYY         dd    100
      palinmax    dd    0
      txtbuf      db    16 dup(0)

.code

start:
      mov    eax,XXX
      mul    YYY
      mov    ebx,eax
      lea    edi,txtbuf
      mov    esi,edi    ;esi=points to first digit
      call   dw2a
      sub    edi,2      ;edi=back pointing to last digit

;this works the same for either 5 or 6 digits
      lodsb
      cmp    al,[edi]
      jnz    nextone
      dec    edi
      lodsb
      cmp    al,[edi]
      jnz    nextone
      dec    edi
      lodsb
      cmp    al,[edi]
      jnz    nextone
      cmp    ebx,palinmax
      jb     nextone
      mov    palinmax,ebx
```

```
nextone:
      inc   YYY
      cmp   YYY,1000
      jb    start
      inc   XXX
      cmp   XXX,999
      ja    endcalc
      push  XXX
      pop   YYY      ;values less than XXX already checked
      jmp   start

dw2a:
      mov   ecx,10
      pushd 0           ;for later use as terminating 0
   @@:
      xor   edx,edx     ;clear for division
      div   ecx
      add   dl,"0"      ;convert to ascii
      push  edx         ;save each digit on stack
      .if   eax != 0
            jmp   @B    ;continue conversion
      .endif
   @@:
      pop   eax         ;retrieve each ascii character
      stosb
      or    al,al
      jnz   @B          ;continue until terminating 0
      ret
endcalc:
```

✓

Tue, 17 Jan 2006, 19:32
gauchopuro
Haskell

Quote Report 5

Another Haskell solution, with GHCi:

```
> let palindrome x = let digits = show x in
                      digits == reverse digits
> maximum [x * y | x <- [100..999], y <- [100..999],
                   palindrome (x * y)]
906609
```

✓

Sun, 25 Mar 2007, 02:10
Vaste
Haskell

Quote Report 5

Fun to see how similar all the Haskell solutions are.

Haskell **Show Code**

```
let isPal x = x == reverse x
maximum $ filter (isPal.show) $ [x*y|x<-[1..999],y<-[1..999]]
```

✓

Sun, 14 Jan 2007, 07:16
Mr.Wizard
Mathematica

Quote Report 5

A solution with Mathematica:

```
pQ=Boole[#==Reverse@#]&@IntegerDigits@#&

Array[pQ[1##]##&,{100,100},900,Max]
```

✓

Post Reply