Problem 39 - Integer Right Triangles

Gautam Manohar

24 June 2018

This document originally appeared as a blog post on my website. Find it at gautammanohar.com/euler/39.

1 Problem Statement

If p is the perimeter of a right triangle with side lengths $\{a, b, c\}$, then there are exactly 3 solutions for $p = 120$:

$$\{20, 48, 52\}, \{24, 45, 51\}, \{30, 40, 50\}$$

For which value of $p \leq N$ is the number of solutions maximized? If there are multiple possible answers, print the minimum one.

2 My Algorithm

A famous method of generating Pythagorean triples is due to Euclid:

$$a = m^2 - n^2 \quad b = 2mn \quad c = m^2 + n^2 \quad m > n \tag{1}$$

It is easy to verify that such choices of a, b, c always form a Pythagorean triple. Furthermore, (1) generates all primitive Pythagorean triples (triples for which $\gcd(a, b, c) = 1$) for coprime m, n of opposite parity. Every Pythagorean triple can be written as $\{ak, bk, ck\}$, where k is some natural number and $\{a, b, c\}$ is a primitive Pythagorean triple.

Our strategy is to generate all primitive Pythagorean triples with perimeter less than N. We can take their multiples to get all Pythagorean triples with perimeter less than N.

When m or n is 0, the triangle produced is an isoceles right triangle. Because such a triangle would have a hypotenuse that is a multiple of $\sqrt{2}$, we know it will not produce a Pythagorean triple. And so $m, n \leq 1$. The perimeter of the triangle is

$$a + b + c = 2m^2 + mn = 2m(m + n). \tag{2}$$

We know $n > 0$, so (2) is at least $2m^2$. This means that we need not search m for which $2m^2 > N_{\max}$. And so we search $1 \le m \le \sqrt{\frac{N_{\max}}{2}}$. We also know that $m > n$ (otherwise a would not be a positive length), so we only search $1 \le n < m$.

For each m, n, we must check three things. First, we check that the perimeter $2m(m+n)$ is not more than N. Then we check that m and n are coprime ($\gcd(m, n) = 1$). Finally, we must check that m and n have opposite parity. This means one of them is odd, the other even. So their sum must be odd. We perform $O(\sqrt{N})$ iterations in each loop for m and n. And so we check each m, n in $O(N)$ time.

If we pass all of these tests, then we have a primitive Pythagorean triple. We will maintain a list `pythag` for which `pythag[i]` gives the number of solutions for perimeter i. For each primitive Pythagorean triple, we increment each `pythag`↩ `[k*P]`, where P is the perimeter of the triple and $1 \le k \le \lfloor \frac{N_{\max}}{P} \rfloor$. This means we count each primitive triple and all the multiples derived from it. Because we make a list, we can answer each query quickly.

We maintain a second list `freq` of the indices corresponding to the strictly right maximal values of `pythag`. That is, for each i in `freq`, we have `pythag[i]` $>$ `pythag[j]` for each $0 \le j < i$. Because this order is strict, we only store the lowest solution in cases where there are multiple. Then, we can binary search `freq` for the largest element less than each query. This is better than slicing `pythag` and using `max`, which is linear in `len(freq)`; binary search is logarithmic. The list `freq` has at most N elements—this is where each $n < N$ is strictly right maximal. This means we answer each query in $O(\log N)$ time. And so our solution has time complexity $O(N_{\max} + T \log N_{\max})$, where T is the number of queries.