Counting Tetrahedra with Integer Sides

Phil Benjamin & Kristen Lew May 4, 2012

Introduction

The original problem was to count the number of non-degenerate non-isomorphic tetrahedra with integer edges and total edge length ***n***. The approach taken is outlined here[1]:

- Generate all sequences of six-positive integers with sum ***n***. These sequences represent the edges of the tetrahedron in the order $[e_{21}, e_{31}, e_{32}, e_{41}, e_{42}, e_{43}]$ where the vertices of the tetrahedron are $\{1, 2, 3, 4\}$.
- Eliminate edge-sequences that violate the triangle inequality
- Eliminate edge-sequences that cannot represent a tetrahedron due to the infeasibility of three-dimensional embedding. (For example, [2, 2, 3, 3, 2, 2] is such a sequence.) This check uses the formula for the volume of a tetrahedron from its edges.
  (See Wikipedia: Tetrahedron – Tartaglia's formula)
- Use Redfield-Pólya enumeration to account for duplicate counting

This approach is implemented in the program CountAllTetrahedra. It was found that the sequence generated was not in the Online Encyclopedia of Integer Sequences. It has now been added as sequence **A208454**. The sequence for $n = 6..30$ is shown here:

					1	0	0	1	1
1	3	2	3	6	6	7	12	11	18
21	25	31	38	46	56	66	76	90	117

This sequence may have some interesting properties. However, there are some limitations:

- The inefficiency in the algorithm used to generate tetrahedra precludes the calculation of a large number of cases. This limits purely numerical investigation of potential patterns.
- The data generated so far, does not indicate a polynomial or C-finite sequence.
- Although it would be nice if there were a simple formula (explicit or recursive), such a formula may not exist for this problem.

With these limitations in mind, we decided to pursue a different approach that we hoped would provide more insight. At this point, it is useful to review what is known about calculating the number of non-degenerate, non-isomorphic triangles with integer sides and integer perimeter.

[1] All Maple code for this investigation is included in the Appendix.

Counting Tetrahedra with Integer Sides

Phil Benjamin & Kristen Lew May 4, 2012

Triangles with Integer Sides

A simple program to count triangles with integer sides and perimeter ***n*** is CountTriangles(n). Here is the count of such triangles for $n = 1..60$.

n												
1 – 12	0	0	1	0	1	1	2	1	3	2	4	3
13 – 24	5	4	7	5	8	7	10	8	12	10	14	12
25 – 36	16	14	19	16	21	19	24	21	27	24	30	27
37 – 48	33	30	37	33	40	37	44	40	48	44	52	48
49 – 60	56	52	61	56	65	61	70	65	75	70	80	75

Note that there is no simple polynomial function that counts these triangles for a given value of ***n***. However, if we consider these values for ***n*** modulo ***12*** an interesting pattern appears:

$n\ mod\ 12$	Formula (x 48)
0	n^2
1	$n^2 + 6n - 7$
2	$n^2 - 4$
3	$n^2 + 6n + 21$
4	$n^2 - 16$
5	$n^2 + 6n - 7$
6	$n^2 + 12$
7	$n^2 + 6n + 5$
8	$n^2 - 16$
9	$n^2 + 6n + 9$
10	$n^2 - 4$
11	$n^2 + 6n + 5$

This shows that if we consider the count for each residue modulo ***12***, a simple polynomial occurs. We can make a further simplification by considering groups of residues modulo ***12***.

Experimental Math Theorem 1[2]: Let $Tri(n)$ count the number of non-degenerate non-isomorphic triangles with integer sides and perimeter n. Let $Tri6(n) = Tri(n) + Tri(n-1) + \cdots + Tri(n-5)$. Let $n \equiv 0\ mod\ 6$. Then $Tri6(n) = \frac{1}{8}(n^2 - 2n)$.

Example: $Tri6(6) = Tri(6) + Tri(5) + Tri(4) + Tri(3) + Tri(2) + Tri(1)$

$$= 1 + 1 + 0 + 1 + 0 + 0 = 3 = \frac{1}{8}(6^2 - 2 \cdot 6)$$

[2] For this study, "Experimental Math Theorem" is used to denote an assertion that appears to be correct based on observed data but for which no proof is presented here.

In other words, by considering the count of triangles with appropriate modulus and then "smoothing" by counting groups of triangles with nearby modulus, we find a nice polynomial formula. We have found other interesting formulas that are not reported here. This theorem is useful for providing an order of magnitude estimate for the number of triangles

Corollary 1.1: $Tri(n) = O(\frac{n^2}{48})$

What is the effect of the triangle inequality on the count of triangles?

Experimental Math Theorem 2: Let $PseudoTri(n)$ count the number of non-degenerate non-isomorphic pseudo-triangles[3] with integer sides and perimeter n.
Let $PseudoTri6(n) = PseudoTri(n) + PseudoTri(n-1) + \cdots + PseudoTri(n-5)$.
Let $n \equiv 0\ mod\ 6$. Then $PseudoTri6(n) = \frac{1}{2}(n^2 - 5n + 8)$.

Corollary 2.1: $PseudoTri(n) = O(\frac{n^2}{12})$

Thus the number of "true" triangles is about one quarter the number of pseudo-triangles.

We consider one more variation for triangles: scalene triangles. Most triangles are scalene. This should be interpreted as meaning that the number of scalene triangles with a given perimeter (integer edges only) is an order of magnitude greater than either the number of equilateral or isosceles triangles. We verify this numerically using the programs CountScalenePseudoTriangles and CountScaleneTriangles.

Experimental Math Theorem 3: Let $ScalenePseudoTri(n)$ count the number of non-degenerate non-isomorphic scalene pseudo-triangles with integer sides and perimeter n.
Let $ScalenePseudoTri6(n) =$

$ScalenePseudoTri(n) + ScalenePseudoTri(n-1) + \cdots + ScalenePseudoTri(n-5)$.
Let $n \equiv 0\ mod\ 6$. Then $PseudoTri6(n) = \frac{1}{2}(n^2 - 11n + 32)$.

Thus we only "lose" a linear number of pseudo-triangles by restricting our attention to the scalene case. Similarly:

What have we learned by considering the problem of counting integer triangles?

- Consider the count of triangles with a given perimeter modulo different numbers
- Count special cases of triangles (equilateral, isosceles, scalene) separately to look for patterns

[3] A pseudo-triangle (here) is a set of three edges that may or may not satisfy the triangle inequality.

Counting Tetrahedra with Integer Sides

Phil Benjamin & Kristen Lew May 4, 2012

- Count pseudo-triangles in order to bound the total number of triangles and to see the effect of the triangle inequality on the count of triangles. We noted that the number of "true" triangles is about 1/4 the number of pseudo-triangles.
- Smooth the counts by grouping perimeters together to look for aggregate numbers

Counting Tetrahedra

We begin by demonstrating the counting sequence used to generate OEIS sequence **A208454**. First consider the count of non-isomorphic pseudo-tetrahedra. This is generated using program CountPseudoTetrahedra. This program uses Redfield-Pólya enumeration. The sequence for $n = 6..30$ is shown here:

					1	1	3	6	11
18	32	48	75	111	160	224	313	420	562
738	956	1221	1550	1936	2405	2958	3609	4368	5260

Next we add the geometric constraints by only considering tetrahedra with positive volume as described above. This count uses the program CountAllTetrahedra.

In order to proceed further, we need a more efficient way to generate the count for tetrahedra. Using our experience with triangles, we define

Definition: a scalene tetrahedron is one with no two equal edges.
(similarly for a scalene pseudo-tetrahedron)

We can produce an efficient enumeration of scalene tetrahedra by considering a canonical representation of the edges of the tetrahedron.

Definition: the normal form for the edge-list for a scalene tetrahedron (e21, e31, e32, e41, e42, e43) satisifies the inequalities: $e_{21} = \min\{e_{ij}\}$ and $e_{31} = \min(\{e_{31}, e_{32}, e_{41}, e_{42}\})$.

In other words e_{21} is the shortest edge. Since e_{43} is the edge "opposite" e_{21}, its identity is now fixed. By choosing e_{31} as the shortest remaining edge, the other edges are determined.

We can use the normal form for scalene tetrahedra to efficiently generate a larger list for investigation without the use of Redfield-Pólya counting. This is accomplished using programs CountNormalTetrahedra and CountNormalPseudoTetrahedra.

Counting Tetrahedra with Integer Sides

Phil Benjamin & Kristen Lew May 4, 2012

The counts generated by CountNormalPseudoTetrahedra are shown here starting with $n = 21$:

30	30	60	90	150	210	330	420	600	780
1050	1320	1740	2130	2700	3300	4080	4890	5970	7050
8460	9930	11730	13620	15960	18360	21270	24330	27930	31710
36180	40800	46200	51870	58350	65160	72960	81060	90270	99930

We noticed that all of the counts in this table are divisible by 30. This is due to the fact that for every set of six distince edge lengths, 30 non-isomorphic scalene pseudo tetrahedra are constructable. Hence this problem can be seen as counting the number of partitions of ***n*** into ***6*** distinct parts. We then tried to extend this table using the program CountPartitionSixDistinct. We were able to extend this count up to $n = 240$. However, we did not discover a polynomial description even using smoothing values up to ***60***. The full list for this calculation is provided in the Appendix.

Our next investigation uses by CountNormalTetrahedra. This counts all scalene tetrahedra with all geometric checks in place. Here, too, we did not find a smoothing function that produced a low degree polynomial. The full list for this calculation is provided in the Appendix.

Although we have not found a simple desription for the count of normal tetrahedra, we did observe the following:

- For Normal Pseudo-Tetrahedra, using the measure $\log_2(NPT[2n]/NPT[n])$ suggests that the sequence is likely approximately a polynomial of degree 5.
- For Normal Tetrahedra, using the measure $\log_2(NT[2n]/NT[n])$ suggests that the sequence is likely approximately a polynomial of degree 5.
- Comparing Normal Pseudo-Tetrahedra and Normal Tetrahedra suggests that about 1% of Pseudo-Tetrahedra are true Tetrahedra.

Counting Tetrahedra with Integer Sides

Phil Benjamin & Kristen Lew May 4, 2012

Appendix: Source Code and Data

```
# Experimental Math Project: Counting Tetrahedra
# Phil Benjamin & Kristen Lew

# This file contains Maple programs used in counting tetrahedra
# with integral sides.

HelpTetra := proc()
        print(` TriangleConstraints(a,b,c) `);
end:

# All needed Maple packages are listed here
with(combinat):
with(LinearAlgebra):

##############################################################################
# Utility programs for sequences
# Phil & Kristen

# Find interpolating polynomial for sequence

# Input:
#       L: list
#       x: symbol for polynomial
#       n0: interpretation of list: a(n0) = L[1], etc.
#       d: difference of arguments: L = [a(n0),a(n0+d),a(n0+2*d),...]
seq2poly := proc(L,x,n0,d)
        local P, a, eq, var, DEG;

        DEG := nops(L) - 1;
        P := add(a[i]*x^i,i=0..DEG);
        var := {seq(a[i],i=0..DEG)};
        eq := {seq(L[1+i] = subs(x=n0+i*d,P),i=0..DEG)};
        var := solve(eq,var);
        subs(var,P);
end:

# Input:
#       L: list
#       si: smoothing interval (number of terms added)
#       d: difference in smoothing intervals
```

```
# Output:
#       smoothed list:
#       SL[i] := add(L[j],j=(1+d*(i-1))..(1+d*(i-1)+si-1)])
smoothList := proc(L,si,d)
        local i, j, t, SL;
        SL := [];
        for i from 1 while 1 + d*(i-1)+si-1 <= nops(L) do
                t := add(L[j],j=(1+d*(i-1))..(1+d*(i-1)+si-1));
                SL := [op(SL),t];
        end;
        return SL;
end:

# Input:
#       L: list
# Output:
#       [d,poly]
#       when smoothing by "si" produces poly of degree < max
seq2goodPoly := proc(L,x,n0)
        local d, si, LS, P;

        for si from 1 to nops(L) do
                for d from 1 to si do
                        LS := smoothList(L,si,d);
                        P := seq2poly(LS,x,n0+si-1,d);
                        if degree(P,x) < nops(LS) - 1 then
                                return [si,d,P];
                        end;
                end;
        end;
        return FAIL;
end:

##############################################################################
# Triangle Investigation

# TriangleConstraints(a,b,c)
# Input: (symbolic) edges of triangle
# Output: set of positive constraints satisfied by edges

TriangleConstraints := proc(a,b,c)
        return {a+b-c,a-b+c,-a+b+c}
```

```
end:

# isTriangle(a,b,c): return true if (a,b,c) satisfies
#       triangle inequality
isTriangle := proc(a,b,c)
        return evalb(min(TriangleConstraints(a,b,c)) > 0);
end:

# isIsosceles(a,b,c): return true if (a,b,c) if nops({a,b,c})<3
isIsosceles := proc(a,b,c)
        return evalb(nops({a,b,c}) < 3);
end:

# AllTriangleConstraints(e21,e31,e32,e41,e42,e43)
# Input: (symbolic) edges of tetrahedron
# Output: set of positive constraints satisfied by edges
#       due to Triangle Constraints

AllTriangleConstraints := proc(e21,e31,e32,e41,e42,e43)
        return TriangleConstraints(e21,e31,e32)
         union TriangleConstraints(e21,e41,e42)
         union TriangleConstraints(e31,e41,e43)
         union TriangleConstraints(e32,e42,e43);
end:


# Count all triangles with integer sides and given perimeter (n)
CountTriangles := proc(n)
        local a,b,c,t;
        t := 0;
        for a from 1 while 3*a <= n do
                for b from a while a + 2*b <= n do
                        c := n - a - b;
                        if isTriangle(a,b,c) then
                                t := t + 1;
                        end if;
                end;
        end;
        return t;
end:

# Count all pseudo-triangles with integer sides and given perimeter (n)
```

```
# That is, do not consider the triangle inequality in the count.
CountPseudoTriangles := proc(n)
        local a,b,c,t;
        t := 0;
        for a from 1 while 3*a <= n do
                for b from a while a + 2*b <= n do
                        c := n - a - b;
                        if true then
                                t := t + 1;
                        end if;
                end;
        end;
        return t;
end:

# Count isosceles triangles with integer sides and given perimeter (n)
CountIsoscelesTriangles := proc(n)
        local a,b,c,t;
        t := 0;
        for a from 1 while 3*a <= n do
                for b from a while a + 2*b <= n do
                        c := n - a - b;
                        if isTriangle(a,b,c) and isIsosceles(a,b,c) then
                                t := t + 1;
                        end if;
                end;
        end;
        return t;
end:


# Count isosceles pseudo-triangles with integer sides and given perimeter (n)
CountIsoscelesPseudoTriangles := proc(n)
        local a,b,c,t;
        t := 0;
        for a from 1 while 3*a <= n do
                for b from a while a + 2*b <= n do
                        c := n - a - b;
                        if isIsosceles(a,b,c) then
                                t := t + 1;
                        end if;
                end;
```

```
        end;
        return t;
end:

# Count scalene pseudo-triangles with integer sides and given perimeter (n)
CountScalenePseudoTriangles := proc(n)
        local a,b,c,t;
        t := 0;
        for a from 1 while 3*a+3 <= n do
                for b from a+1 while a + 2*b + 1 <= n do
                        c := n - a - b;
                        t := t + 1; # no geometric check
                end;
        end;
        return t;
end:

# Count scalene triangles with integer sides and given perimeter (n)
CountScaleneTriangles := proc(n)
        local a,b,c,t;
        t := 0;
        for a from 1 while 3*a+3 <= n do
                for b from a+1 while a + 2*b + 1 <= n do
                        c := n - a - b;
                        if isTriangle(a,b,c) then
                                t := t + 1;
                        end if;
                end;
        end;
        return t;
end:


################################################################################
# Tetrahedron counting programs
# Phil & Kristen

# Volume of Tetrahedron using Cayley-Menger determinant
# (a.k.a. Tartaglia's formula)
# Note: this formula computes 288 * Volume^2 however we are
# only checking for positive values.  For some special cases
# these values can be factored to produce faster checks
```

Counting Tetrahedra with Integer Sides

Phil Benjamin & Kristen Lew May 4, 2012

```
TetraVolume := proc(e21,e31,e32,e41,e42,e43)
        local M;
        M := Matrix([
                [0,   1,   1,   1,   1],
                [1,   0,e21^2,e31^2,e41^2],
                [1,e21^2,   0,e32^2,e42^2],
                [1,e31^2,e32^2,   0,e43^2],
                [1,e41^2,e42^2,e43^2,   0]
        ]);
        return Determinant(M);
end:

# In order to achieve a speedup in Tetrahedron enumeration
# FastTetraVolume "hard codes" the volume calculation above

FastTetraVolume := proc(a,b,c,d,e,f)
        return
          - a^2*f^4 - b^2*e^4 - d^2*c^4 - b^4*e^2 - d^4*c^2 - a^4*f^2
          - c^2*f^2*e^2 - b^2*f^2*d^2 - a^2*e^2*d^2 - a^2*c^2*b^2
          + b^2*f^2*e^2 + d^2*c^2*f^2 + a^2*f^2*e^2 + d^2*e^2*c^2
          + a^2*c^2*f^2 + b^2*e^2*c^2 + b^2*a^2*f^2 + b^2*e^2*d^2
          + d^2*a^2*f^2 + d^2*c^2*b^2 + a^2*e^2*b^2 + d^2*a^2*c^2;
end:

# Check feasibility of tetrahedron by checking triangle inequality for all sides
# and checking for positive volume

isTetrahedron := proc(e21,e31,e32,e41,e42,e43)
        if isTriangle(e21,e31,e32) and isTriangle(e21,e41,e42)
          and isTriangle(e31,e41,e43) and isTriangle(e32,e42,e43)
          and FastTetraVolume(e21,e31,e32,e41,e42,e43) > 0 then
                return true;
        end if;
        return false;
end:

# Edge permutation group for tetrahedra
# In order to check for isomorphic tetrahedra, check all permutations
# of the edges.  This program finds all edge permutations generated
# by vertex permutations
```

```
EdgePermGroup := proc()
        local M, M2, L, L2, Lset, Perm, AllPerm, i, j;
        # Edges are listed in order: e21 e31 e32 e41 e42 e43
        L := [1,2,3,4,5,6];
        Lset := {L};
        M := Matrix([[0,0,0,0],[1,0,0,0],[2,3,0,0],[4,5,6,0]]);
        M := M + Transpose(M); # for symmetry

        # Apply all permutations to the indices of M
        AllPerm := permute(4);
        for Perm in AllPerm do
                M2 := Matrix(4,4); # initialize M2
                for i from 1 to 4 do
                        for j from 1 to 4 do
                                M2[i,j] := M[Perm[i],Perm[j]];
                        end;
                end;
                L2 := [M2[2,1],M2[3,1],M2[3,2],M2[4,1],M2[4,2],M2[4,3]];
                Lset := Lset union {L2};
        end;
        return Lset;
end:

# Count edge permutations of given list of edges by applying each
# member of EdgePermGroup to the edges to determine how many distinct
# edge sequences are formed
CountEdgePerm := proc(PermGroup,EdgeList)
        local Perm, EdgeList2, EdgeListSet, i;
        EdgeListSet := {EdgeList};
        for Perm in PermGroup do
                # apply permutation to the EdgeList
                EdgeList2 := [seq(EdgeList[Perm[i]],i=1..nops(Perm))];
                EdgeListSet := EdgeListSet union {EdgeList2};
        end;
        return nops(EdgeListSet);
end:

# Count non-equivalent pseudo-tetrahedra with given edge sum
# (no geometric constraints imposed)
CountPseudoTetrahedra := proc(n)
        local Count, Gp, e21, e31, e32, e41, e42, e43, eList;
        Count := 0;
```

```
        Gp := EdgePermGroup();
        for e21 from 1 while e21 + 5 <= n do
                for e31 from 1 while e21 + e31 + 4 <= n do
                        for e32 from 1 while e21 + e31 + e32 + 3 <= n do
                                for e41 from 1 while e21 + e31 + e32 + e41 + 2 <= n do
                                        for e42 from 1 while e21 + e31 + e32 + e41 + e42 + 1 <= n do
                                                e43 := n - e21 - e31 - e32 - e41 - e42;
                                                eList := [e21,e31,e32,e41,e42,e43];
                                                Count := Count + 1/CountEdgePerm(Gp,eList);
                                        end;
                                end;
                        end;
                end;
        end;
        return Count;
end:

# Count all non-equivalent tetrahedra with given edge sum
# Geometric feasibility is checked using the TetraVolume program
CountAllTetrahedra := proc(n)
        local Count, Gp, e21, e31, e32, e41, e42, e43, eList;
        Count := 0;
        Gp := EdgePermGroup();
        for e21 from 1 while e21 + 5 <= n do
                for e31 from 1 while e21 + e31 + 4 <= n do
                        for e32 from 1 while e21 + e31 + e32 + 3 <= n do
                                for e41 from 1 while e21 + e31 + e32 + e41 + 2 <= n do
                                        for e42 from 1 while e21 + e31 + e32 + e41 + e42 + 1 <= n do
                                                e43 := n - e21 - e31 - e32 - e41 - e42;
                                                if not isTetrahedron(e21,e31,e32,e41,e42,e43) then
                                                        next;
                                                end if;
                                                eList := [e21,e31,e32,e41,e42,e43];
                                                Count := Count + 1/CountEdgePerm(Gp,eList);
                                        end;
                                end;
                        end;
                end;
        end;
        return Count;
end:
```

Counting Tetrahedra with Integer Sides

Phil Benjamin & Kristen Lew May 4, 2012

```
# Count all non-equivalent scalene pseudo-tetrahedra with given edge sum
# Geometric feasibility is not checked.
# Use normal form instead of Redfield-Polya counting
CountNormalPseudoTetrahedra := proc(n)
        local Count, e21, e31, e32, e41, e42, e43;
        Count := 0;
        for e21 from 1 while 6*e21 + 15 <= n do
                for e31 from e21 + 1 while e21 + e31 + 4*e21 + 10 <= n do
                        for e43 from e21 + 1 while e21 + e31 + e43 + 3*e31+6 <= n do
                                if e43 = e31 then next; end if;
                                for e32 from e31 + 1 while e21 + e31 + e43 + e32 + 2*e31 + 3 <= n do
                                        if e32 = e43 then next; end if;
                                        for e41 from e31 + 1 while e21 + e31 + e43 + e32 + e41 + 1 <= n
do
                                                if e41 = e43 or e41 = e32 then next; end if;
                                                e42 := n - e21 - e31 - e32 - e41 - e43;
                                                if e42 <= 0 then
                                                        print("OOPS:");
print([e21,e31,e32,e41,e42,e43]);
                                                        return FAIL;
                                                end if;
                                                if e42 <= e21 or e42 <= e31
                                                  or e42 = e43 or e42 = e32 or e42 = e41 then next; end
if;
                                                Count := Count + 1;
                                        end;
                                end;
                        end;
                end;
        end;
        return Count;
end:

# Check validity of CountScalenePseudoTetrahedra using an integer partition argument
# Count partitions of n using distinct parts
# CountScalenePseudoTetrahedra should be 30 times this number
CountPartitionSixDistinct := proc(n)
        local Count,a,b,c,d,e,f;
        Count := 0;
        for a from 1 while 6*a + 15 <= n do
          for b from a+1 while a + 5*b + 10 <= n do
            for c from b+1 while a + b + 4*c + 6 <= n do
```

```
            for d from c+1 while a + b + c + 3*d + 3 <= n do
              for e from d+1 while a + b + c + d + 2*e + 1 <= n do
                f := n - a - b - c - d - e;
                    if f <= e then print("OOPS!"); print(a,b,c,d,e,f); end if;
                    Count := Count + 1;
              end;
             end;
            end;
           end;
          end;
        return Count;
end:


# Count all non-equivalent scalene tetrahedra with given edge sum
# Geometric feasibility is checked.
# Use normal form instead of Redfield-Polya counting
CountNormalTetrahedra := proc(n)
        local Count, e21, e31, e32, e41, e42, e43;
        Count := 0;
        for e21 from 1 while 6*e21 + 15 <= n do
                for e31 from e21 + 1 while e21 + e31 + 4*e21 + 10 <= n do
                        for e43 from e21 + 1 while e21 + e31 + e43 + 3*e31+6 <= n do
                                if e43 = e31 then next; end if;
                                for e32 from e31 + 1 while e21 + e31 + e43 + e32 + 2*e31 + 3 <= n do
                                        if e32 = e43 then next; end if;
                                        for e41 from e31 + 1 while e21 + e31 + e43 + e32 + e41 + 1 <= n
do
                                                if e41 = e43 or e41 = e32 then next; end if;
                                                e42 := n - e21 - e31 - e32 - e41 - e43;
                                                if e42 <= 0 then
                                                        print("OOPS:");
print([e21,e31,e32,e41,e42,e43]);
                                                        return FAIL;
                                                end if;
                                                if e42 <= e21 or e42 <= e31
                                                  or e42 = e43 or e42 = e32 or e42 = e41 then next; end
if;
                                                if not isTetrahedron(e21,e31,e32,e41,e42,e43) then
next; end if;
                                                Count := Count + 1;
                                        end;
```

```
                                    end;
                                end;
                            end;
                        end;
                        return Count;
end:

##############################################################################################
############################
# Data from investigations
# Count of NormalPseudoTetrahedra
NPT := [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 30,
  30, 60, 90, 150, 210, 330, 420, 600, 780, 1050, 1320, 1740,
  2130, 2700, 3300, 4080, 4890, 5970, 7050, 8460, 9930, 11730,
  13620, 15960, 18360, 21270, 24330, 27930, 31710, 36180, 40800,
  46200, 51870, 58350, 65160, 72960, 81060, 90270, 99930, 110760,
  122100, 134820, 148050, 162810, 178260, 195300, 213120, 232800,
  253260, 275760, 299250, 324870, 351600, 380760, 411060, 444000,
  478320, 515400, 554010, 595740, 639030, 685680, 734190, 786210,
  840270, 898230, 958290, 1022550, 1089240, 1160310, 1234020,
  1312560, 1393830, 1480260, 1569810, 1664730, 1763010, 1867170,
  1974810, 2088720, 2206530, 2330850, 2459370, 2594970, 2734920,
  2882370, 3034650, 3194730, 3359970, 3533640, 3712650, 3900570,
  4094370, 4297410, 4506720, 4725960, 4951680, 5187870, 5431140,
  5685270, 5946900, 6220140, 6501150, 6794370, 7096020, 7410300,
  7733490, 8070150, 8416020, 8776020, 9145950, 9530490, 9925500,
  10336020, 10757370, 11194950, 11644140, 12110100, 12588270,
  13084200, 13592730, 14119800, 14660340, 15220020, 15793830,
  16387830, 16996410, 17626050, 18271200, 18938070, 19621170,
  20327130, 21049830, 21796320, 22560570, 23349330, 24156630,
  24989670, 25841820, 26720730, 27619830, 28546470, 29494170,
  30470730, 31468980, 32497170, 33548220, 34630080, 35735730,
  36873600, 38035950, 39231690, 40453170, 41708970, 42991530,
  44309910, 45655800, 47038770, 48450600, 49900530, 51380400,
  52899960, 54450300, 56041680, 57665280, 59331000, 61030110,
  62773050, 64550280, 66372780, 68231100, 70135860, 72077700,
  74067780, 76095930, 78173850, 80291460, 82460100, 84669780,
  86932410, 89237130, 91596420, 93999540, 96458580, 98962890,
  101525130, 104133780, 106802100, 109518660, 112296330,
  115123770, 118014450, 120956130, 123962880, 127022580,
  130148880, 133329750, 136579470, 139885080, 143261490,
  146695830, 150202590, 153769020, 157410240, 161112540,
```

```
  164891670, 168734040, 172654980, 176640990, 180708060,
  184841700, 189058560, 193344270, 197715030, 202156590,
  206685810, 211287420, 215978970, 220745310]:

# Count of (true) NormalTetrahedra
NT := [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
  0, 0, 0, 0, 0, 4, 1, 1, 3, 2, 5, 15, 10, 13, 17, 18, 27, 48,
  41, 53, 57, 81, 89, 118, 124, 146, 175, 212, 232, 285, 302,
  363, 411, 479, 512, 603, 665, 749, 845, 939, 1024, 1182, 1262,
  1452, 1563, 1716, 1898, 2116, 2284, 2526, 2714, 3004, 3229,
  3574, 3828, 4173, 4516, 4904, 5265, 5739, 6110, 6621, 7108,
  7681, 8192, 8835, 9383, 10099, 10801, 11526, 12271, 13169,
  13927, 14956, 15806, 16830, 17858, 19017, 20134, 21396, 22579,
  23984, 25247, 26913, 28259, 29883, 31506, 33301, 35041, 37007,
  38880, 40970, 43033, 45364, 47515, 50041, 52427, 55116, 57719]:
```