Horns frame

https://github.com/heptagons/meccano/frames/horns

Abstract

Horns frame is a group of seven meccano [1] strips intended to build polygons. We found the formula to calculate the internal angles then look for polygons. With sofware we found hexagons, octagons and dodecagons. We found no pentagons.

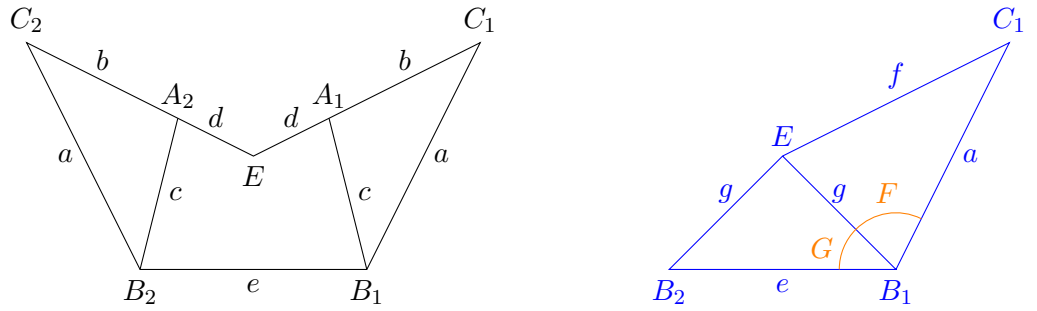

Figure 1: The **horn frame** has seven strips: Two of length a, two of length $b+d$, two of length c and one of length e. We expect to build polygons with internal angle $C_1B_1B_2$ and perimeter including segments a, e, a.

1 Algebra

From figure 1 we start with triangle $\triangle A_1B_1C1$. At vertex A_1 we have angle A and the supplement A':

$$A \equiv \angle B_1A_1C_1 \tag{1}$$

$$\cos A = \frac{b^2+c^2-a^2}{2bc} \qquad \text{if and only if } a < b+c \tag{2}$$

$$A' \equiv \angle EA_1B_1 = \pi - A \tag{3}$$

$$\cos A' = \cos(\pi - A) = -\cos A = \frac{-b^2-c^2+a^2}{2bc} \tag{4}$$

We define $f \equiv b+d$ and $g \equiv \overline{EB_1}$. With the law of cosines we have:

$$f \equiv \boxed{b+d} \qquad \in \mathbb{N} \tag{5}$$

$$g^2 = c^2+d^2-2cd\cos A' \tag{6}$$

$$\begin{aligned} &= c^2+d^2-(2cd)\frac{-b^2-c^2+a^2}{2bc} \\ &= \frac{bc^2+bd^2+b^2d+c^2d-a^2d}{b} \\ &= \frac{(b+d)(bd+c^2)-a^2d}{b} \end{aligned} \tag{7}$$

[1] Meccano mathematics by 't Hooft

Define a new variable $h = (b+d)(bd+c^2) - a^2 d$:

$$h \equiv \boxed{(bd+c^2)f - a^2 d} \qquad \in \mathbb{Z} \tag{8}$$

$$g^2 = \boxed{\frac{h}{b}} \qquad \text{if and only if } 0 < h < b \tag{9}$$

We calculate angles $F \equiv \angle C_1 B 1 E$ and $G \equiv \angle B_2 B_1 E$. We replace g^2 by h/b:

$$\cos F = \frac{a^2 + g^2 - f^2}{2ag} = \frac{a^2 b - bf^2 + h}{2abg} \tag{10}$$

$$\cos G = \boxed{\frac{e}{2g}} \tag{11}$$

Define new variable $j = bf^2 - a^2 b - h$ so:

$$j \equiv \boxed{bf^2 - a^2 b - h} \qquad \in \mathbb{Z} \tag{12}$$

$$\cos F = \frac{a^2 b - bf^2 + h}{2abg} = \boxed{\frac{-j}{2abg}} \tag{13}$$

We calculate cosines squares and products. Again we replace g^2 by h/b:

$$\cos F \cos G = \frac{ej}{4abg^2} = \frac{bej}{4abh} = \boxed{\frac{ej}{4ah}} \qquad \in \mathbb{Q} \tag{14}$$

$$\cos^2 F = \frac{j^2}{4a^2 b^2 g^2} = \frac{bj^2}{4a^2 b^2 h} = \boxed{\frac{j^2}{4a^2 bh}} \qquad \in \mathbb{Q} \tag{15}$$

$$\cos^2 G = \frac{e^2}{4g^2} = \boxed{\frac{be^2}{4h}} \qquad \in \mathbb{Q} \tag{16}$$

$$\cos^2 F \cos^2 G = \frac{be^2 j^2}{16a^2 bh^2} = \boxed{\frac{e^2 j^2}{16a^2 h^2}} \qquad \in \mathbb{Q} \tag{17}$$

$$\tag{18}$$

We calculate the sines part squared and set a common denominator as square $16a^2 b^2 h^2$:

$$\begin{aligned}
(\sin F \sin G)^2 &= (1 - cos^2 F)(1 - cos^2 G) && (19)\\
&= 1 - cos^2 F - cos^2 G + cos^2 F \cos^2 G \\
&= 1 - \frac{j^2}{4a^2 bh} - \frac{be^2}{4h} + \frac{e^2 j^2}{16a^2 h^2} \\
&= 1 - \frac{j^2}{4a^2 bh} - \frac{be^2}{4h} + \frac{e^2 j^2}{16a^2 h^2} \\
&= \frac{16a^2 b^2 h^2 - (4bh)j^2 - (4a^2 b^2 h)be^2 + (b^2)e^2 j^2}{16a^2 b^2 h^2} \\
&= \frac{16a^2 b^2 h^2 - 4bhj^2 - 4a^2 b^3 e^2 h + b^2 e^2 j^2}{16a^2 b^2 h^2} \\
&= \frac{b(be^2 - 4h)(j^2 - 4a^2 bh)}{16a^2 b^2 h^2} && (20)
\end{aligned}$$

Extract square root to get $\sin F \sin G = \sqrt{D}/A$ where $D, A \in \mathbb{Z}$:

$$\sin F \sin G = \boxed{\frac{\sqrt{b(be^2 - 4h)(j^2 - 4a^2 bh)}}{4abh}} \qquad \in \mathbb{A} \tag{21}$$

We sum the angles F and G to get:

$$\begin{aligned}\cos(F+G) &= \frac{ej}{4ah} - \frac{\sqrt{b(be^2-4h)(j^2-4a^2bh)}}{4abh} \\ &= \frac{bej - \sqrt{b(be^2-4h)(j^2-4a^2bh)}}{4abh} \qquad \in \mathbb{A} \end{aligned} \tag{22}$$

2 Software

We write a general routine called `HornsE` to iterate over increasing segments sizes e, a, b, c, d from min to max and filter cosines of the form $B + C\sqrt{D}/A$:

```
func HornsE(min, max N32, found func(a, b, c, d, e N32), den N32, num ...Z32) {
  factory := NewA32s()
  for e := min; e <= max; e++ {
    for a := min; a <= e; a++ {
      ea := NatGCD(e, a)
      for b := min; b <= e; b++ {
        eab := NatGCD(ea, b)
        for c:= min; c <= max; c++ {
          if a >= b+c || b >= a+c || c >= a+b {
            continue // impossible triangle abc
          }
          eabc := NatGCD(eab, c)
          for d := min; d <= max; d++ {
            if eabcd := NatGCD(eabc, d); eabcd > 1 {
              continue // scaled repetition
            }
            f := b + d
            h := (b*d + c*c)*f - a*a*d
            j := -(a*a*b - b*f*f + h) // non zero for hexagons
            na := 4*N(a)*N(b)*N(h)

            zb := Z(b)*Z(e)*Z(j)
            zd0 := Z(b)
            zd1 := Z(b)*Z(e)*Z(e) - 4*Z(h)
            zd2 := Z(j)*Z(j) - 4*Z(a)*Z(a)*Z(b)*Z(h)
            if zd1 == 0 || zd2 == 0 { // hexagon arcos=1/2
              if cos, err := factory.ANew1(na, zb); err != nil {
                // silent overflow
              } else if cos.Equals(den, num...) {
                found(a, b, c, d, e)
              }
            } else if zd := zd0*zd1*zd2; zd < 0 {
              // skip imaginary numbers
            } else if cos, err := factory.ANew3(na, zb, 1, zd); err != nil {
              // silent overflow
            } else if cos.Equals(den, num...) {
              found(a, b, c, d, e)
            }
          }
        }
      }
    }
  }
}
```

3 Polygons

3.1 Hexagons

For hexagons we call function `TestHornsEHexagons` which filters for cosine 1/2. We reject later trivial cases where $a = b = c$ that is, an equilateral triangle present:

```
func TestHornsEHexagons(t *testing.T) {
  min, max := N32(0), N32(20)
  fmt.Printf("segments min=%d max=%d a,b,c,d,e:\n", min, max)
  i := 0
  HornsE(min, max, func(a, b, c, d, e N32) {
    // Interesting hexagons are those when abc triangle is not equilateral
    if a != b && a !=c {
      i++
      fmt.Printf("% 3d) %d,%d,%d,%d,%d\n", i, a, b, c, d, e)
    }
  },2,1) //  cos 60
}
```

For a range of sizes from 0 to 20 we get 16 hexagons, some non trivial:

```
segments min=0 max=20 a,b,c,d,e:
  1) 3,7,5,0,10
  2) 3,8,7,0,13
  3) 10,13,7,0,13
  4) 8,13,7,0,14
  5) 3,8,7,1,15
  6) 5,8,7,2,15
  7) 11,7,7,7,15
  8) 7,13,8,0,16
  9) 3,8,7,2,17
 10) 5,8,7,3,17
 11) 16,14,6,7,17
 12) 8,7,5,7,18
 13) 3,8,7,3,19
 14) 5,8,7,4,19
 15) 7,3,5,11,19
 16) 7,8,5,6,19
--- PASS: TestHornsEHexagons (28.73s)
```

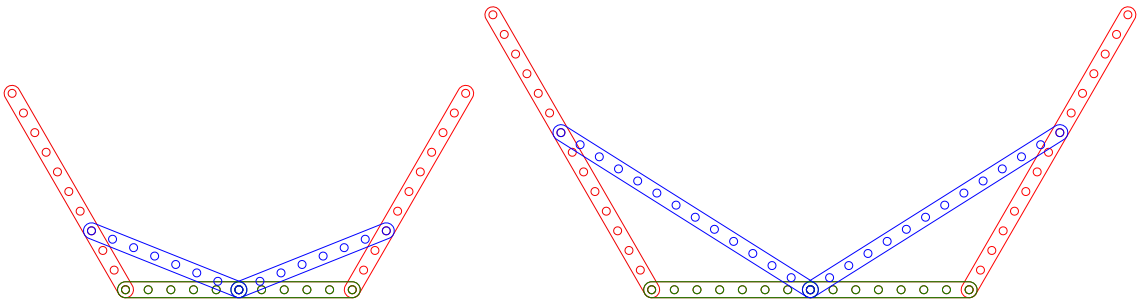

Figure 2: Trivial hexagons of size 10 and 14. $d = 0$. We call the solutions trivial because contain triangles used by simpler hexagons search.

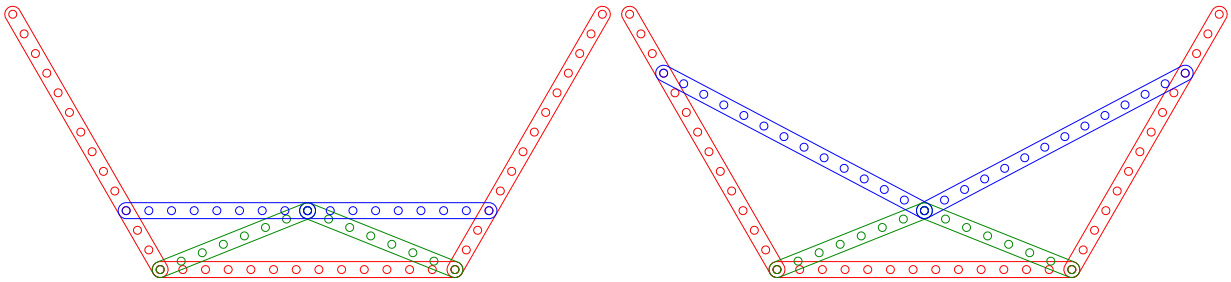

Figure 3: Hexagons of size 13. $d = 0$. Image at the right is not trivial.

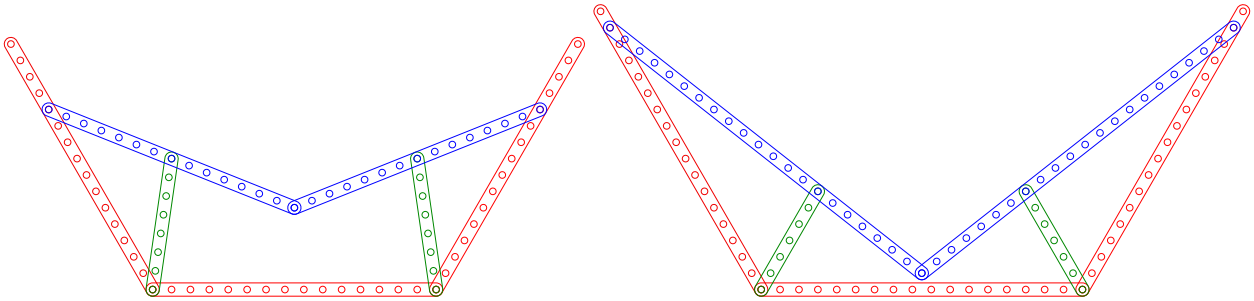

Figure 4: Non trivial hexagons sizes 15 and 17.

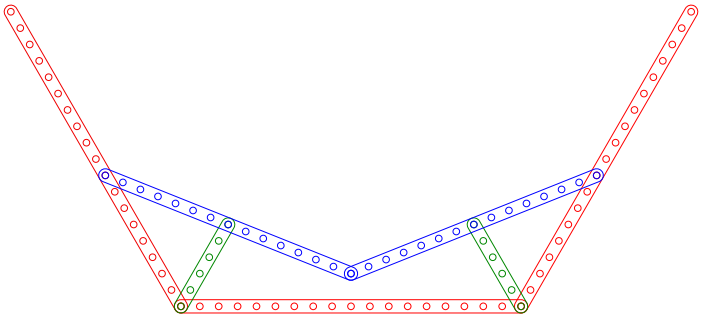

Figure 5: Non trivial hexagon of size 18.

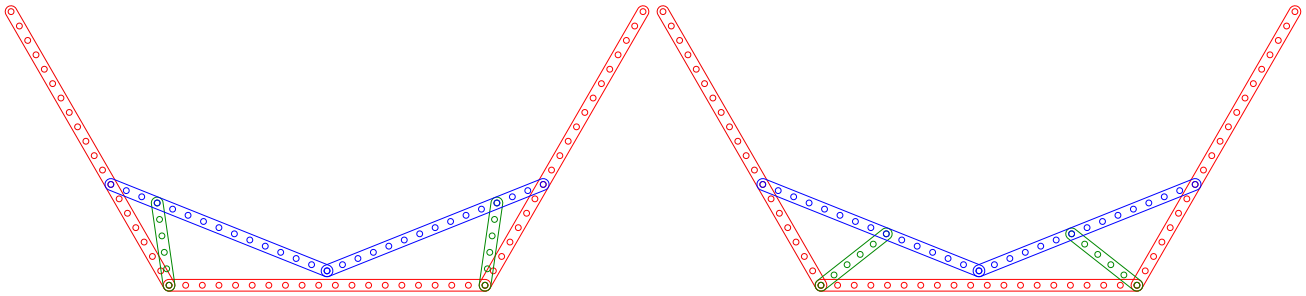

Figure 6: Non trivial hexagons of sizes 19.

3.2 Octagons

For octagons we call function `TestHornsEOctagons` which filters for cosine $\sqrt{2}/2$

```
1 func TestHornsEOctagons(t *testing.T) {
```

```
  min, max := N32(1), N32(40)
  fmt.Printf("segments min=%d max=%d a,b,c,d,e:\n", min, max)
  i := 0
  HornsE(min, max, func(a, b, c, d, e N32) {
    i++
    fmt.Printf("% 3d) %d,%d,%d,%d,%d\n", i, a, b, c, d, e)
  },2,0,1,2) //  cos 45 degrees sqrt{2}/2
}
```

For a range of segments sizes from 1 to 40 we get 10 solutions:

```
segments min=1 max=40 a,b,c,d,e:
  1) 2,3,3,3,8
  2) 3,2,3,7,12
  3) 14,9,9,9,16
  4) 14,11,11,11,24
  5) 21,21,14,6,24
  6) 7,18,17,3,28
  7) 9,10,11,17,36
  8) 9,20,19,7,36
  9) 10,9,11,21,40
 10) 35,27,22,18,40
--- PASS: TestHornsEOctagons (566.86s)
```

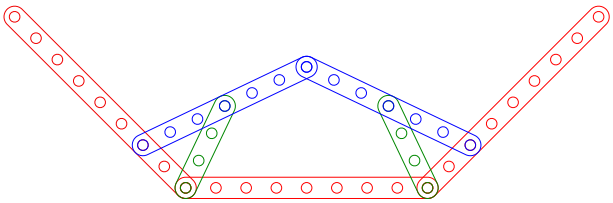

Figure 7: Octagon of size 8.

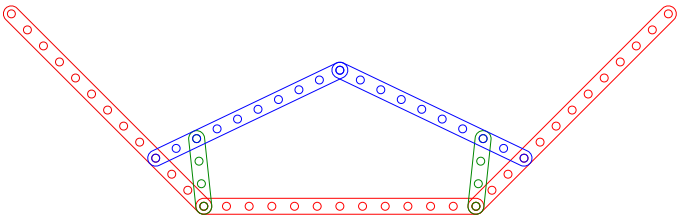

Figure 8: Octagon of size 12.

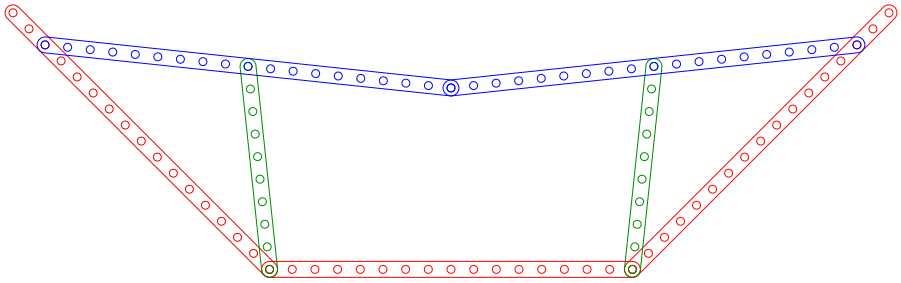

Figure 9: Octagon of size 16.

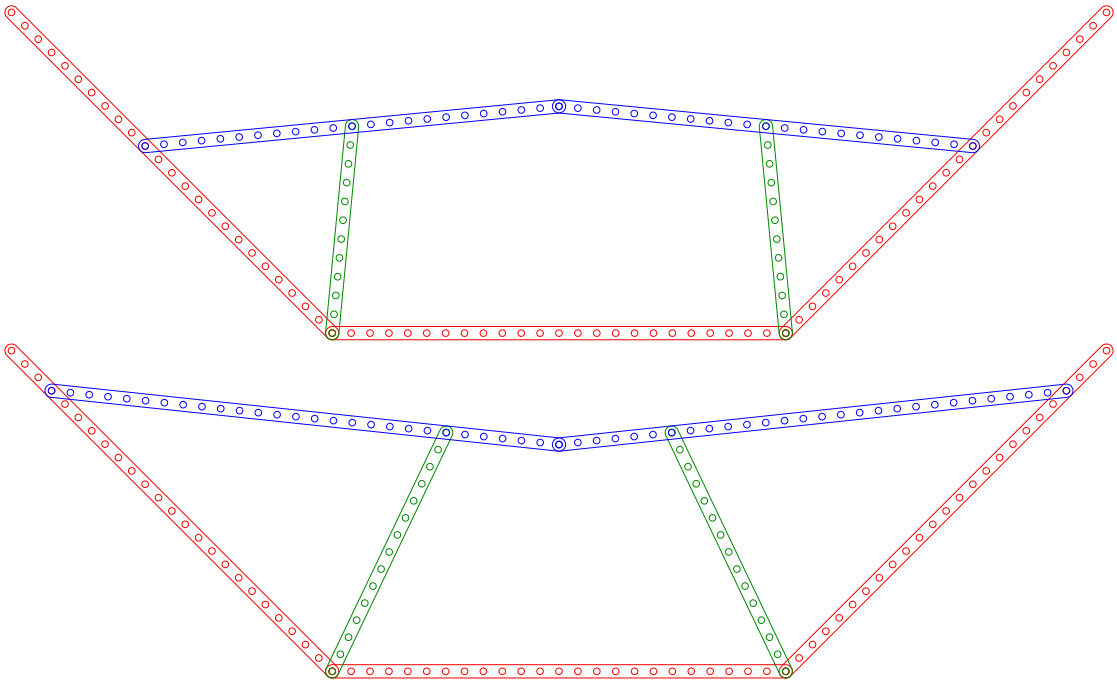

Figure 10: Octagons of sizes 24.

3.3 Dodecagons

For dodecagons we call function `TestHornsEDodecagons` which filters for cosine $\sqrt{3}/2$

```
func TestHornsEDodecagons(t *testing.T) {
  min, max := N32(1), N32(40)
  fmt.Printf("segments min=%d max=%d a,b,c,d,e:\n", min, max)
  i := 0
  HornsE(min, max, func(a, b, c, d, e N32) {
    i++
    fmt.Printf("% 3d) %d,%d,%d,%d,%d\n", i, a, b, c, d, e)
  }, 2,0,1,3) //  cos 30 degrees sqrt{3}/2
}
```

For a range of segments 1 to 40 we found 9 solutions:

```
segments min=1 max=40 a,b,c,d,e:
  1) 6,5,5,5,8
  2) 15,4,13,21,20
  3) 15,9,12,16,20
  4) 15,14,13,11,20
  5) 15,18,15,7,20
  6) 10,13,13,13,24
  7) 21,10,17,25,28
  8) 16,17,17,17,30
  9) 30,11,25,39,40
--- PASS: TestHornsEDodecagons (566.19s)
```

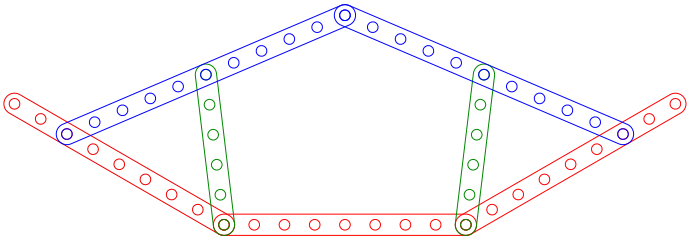

Figure 11: Dodecagon of size 8.

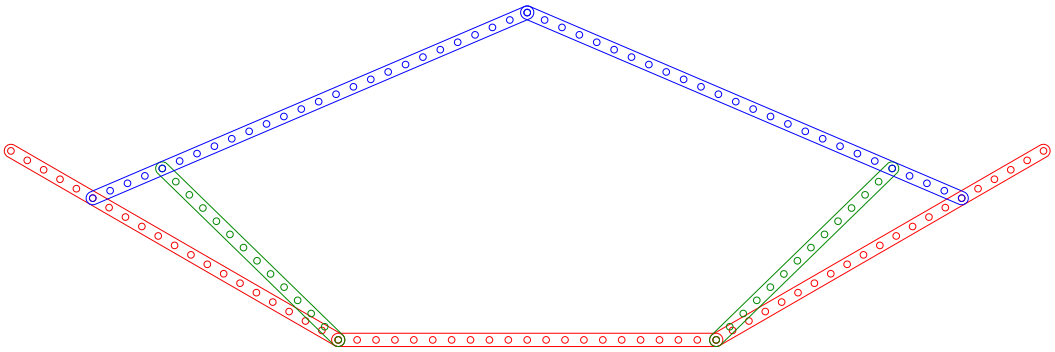

Figure 12: Dodecagon of size 20. Special case $d > e$.

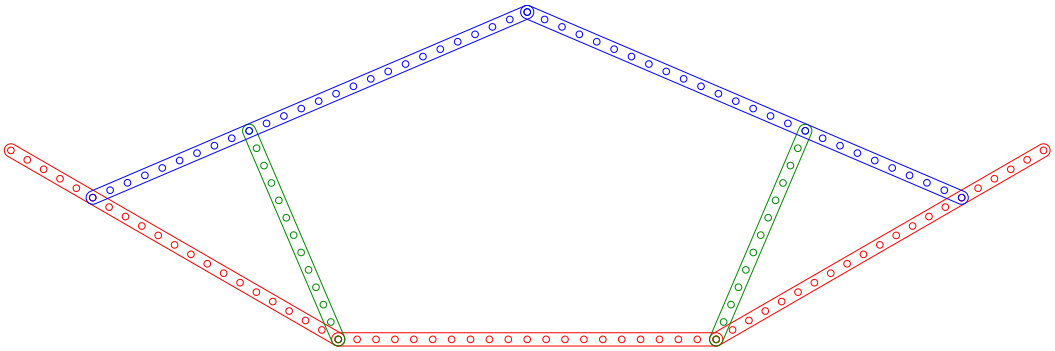

Figure 13: Dodecagon of size 20.

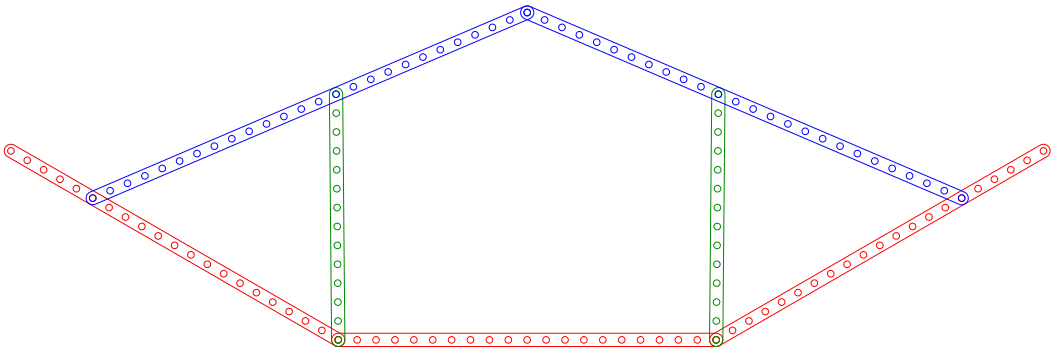

Figure 14: Dodecagon of size 20.

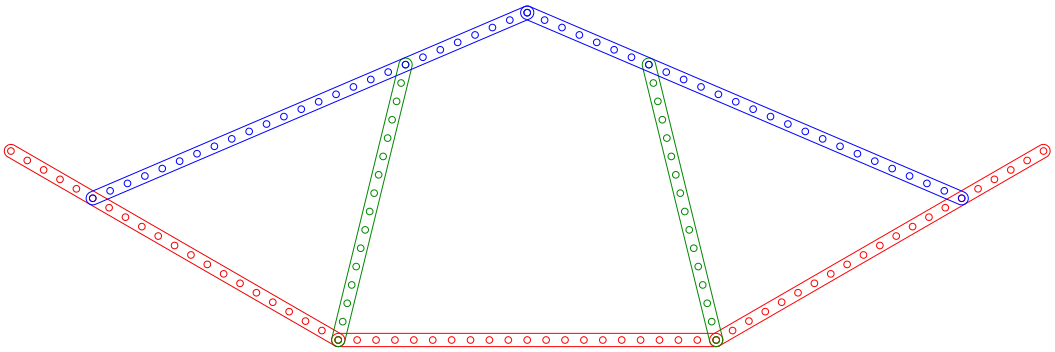

Figure 15: Dodecagon of size 20.

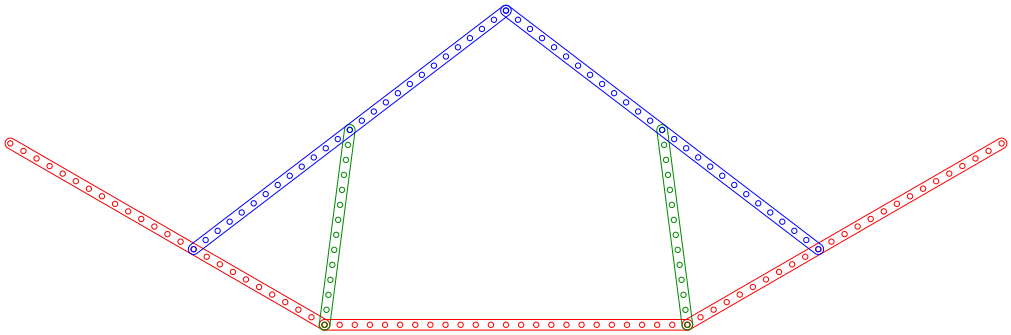

Figure 16: Dodecagon of size 24.

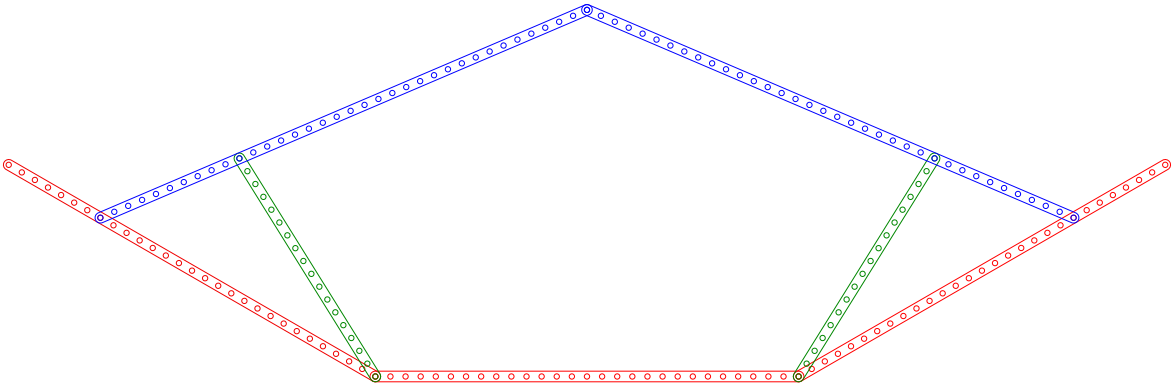

Figure 17: Dodecagon of size 28.

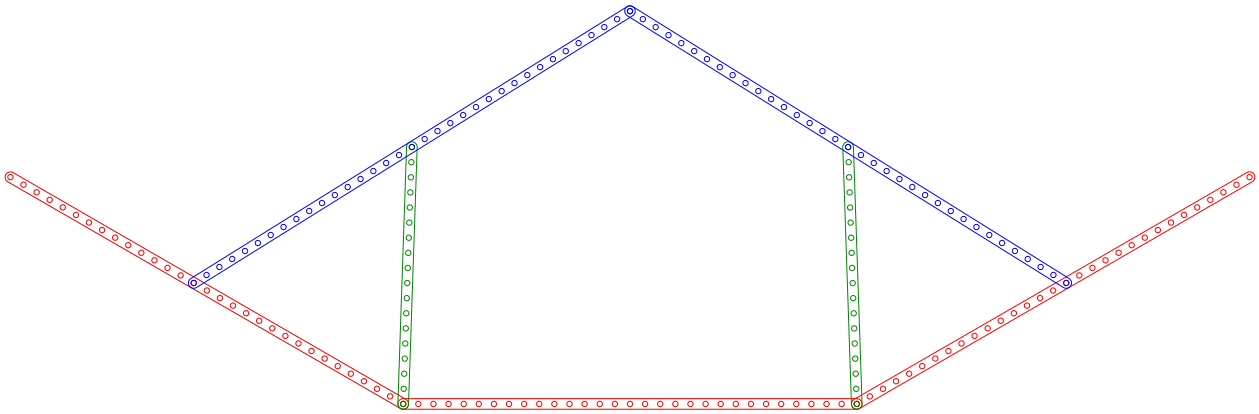

Figure 18: Dodecagon of size 30.