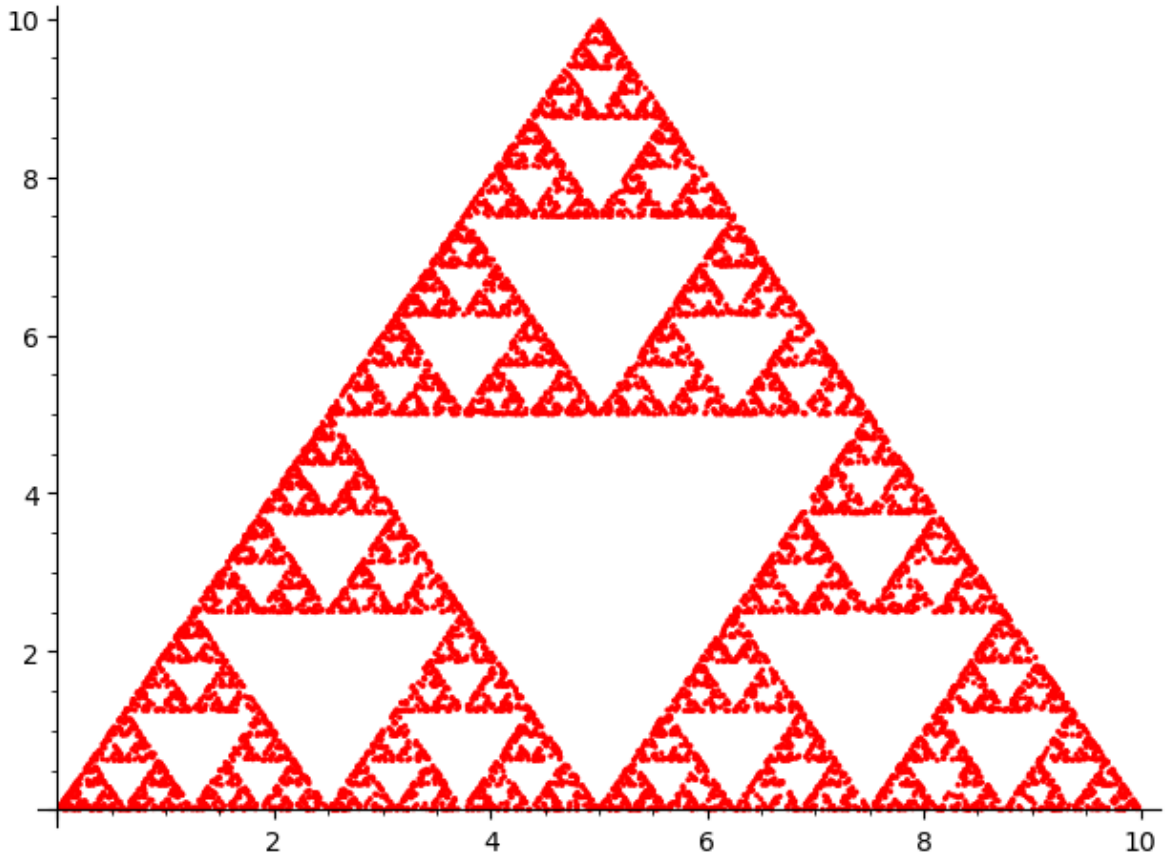
In [6]:
```
# Problem 1

import random

def problem1(L, t, n):
    P = []
    currPos = t
    for i in range(n):
        r = random.randint(1,6)
        if r == 1 or r == 2:
            currPos =( (currPos[0] + L[0][0])/2.0, (currPos[1] + L[0][1])/2.0)
        elif r == 3 or r == 4:
            currPos =( (L[1][0] + currPos[0])/2.0, (currPos[1] + L[1][1])/2.0)
        else:
            currPos =( (L[2][0] + currPos[0])/2.0, (L[2][1] + currPos[1])/2.0)
#         print(r, currPos)
        P.append(currPos)
    Pt = point(P, color='red', size = 5)
    plot(Pt).show()


problem1([(0,0), (10,0), (5,10)], (4,2), 10000)
```



In [7]:
```
# Problem 2
reset()

var('x','y','z','w')
def problem2(f, g, h):
    F(x,y,z,w) = f
    G(x,y,z,w) = g.lhs()
    H(x,y,z,w) = h.lhs()
    gradF = F.diff()
    gradG = G.diff()
    gradH = H.diff()

    var('q1','q2')  #Lambda
    eqs = [gradF(x,y,z,w)[k] - q1*gradG(x,y,z,w)[k] - q2*gradH(x,y,z,w)[k] for
k in range(4)]
    eqs.append(g)
    eqs.append(h)

    sols = solve(eqs, [x,y,z,w,q1,q2])

    result = []
    for i in range(len(sols)):
        result.append(f.subs(sols[i]))
    return max(result)

print(problem2(4*x+4*y+9*z-2*w, 2*x+2*y+z+w==0, x^2+y^2+z-4==0))
print(problem2(x-y+2*z+2*w, x+y+z+w==0, x^2+y^2+z^2-3==0))
```

```
516/11
sqrt(30)
```

In [8]:
```
# Problem 3
def problem3(L):
    numCross = 0
    for i in range(len(L)):
        for j in range(i):
            if (L[i] < L[j]):
                numCross += 1
    return numCross

# print(problem3([3,0,9,4,2,6,1,8,5,7]))
problem3([10,3,13,19,0,11,9,20,4,16,2,12,6,1,14,15,8,5,17,18,7])
```

Out[8]: 97

In [9]:
```
# Problem 4
def distance(a,b):
    length1 = abs(a[0]-b[0])
    length2 = abs(a[1]-b[1])
    result = sqrt(length1^2 + length2^2)
    return numerical_approx(result)

def problem4(f):

    R = QQ[x,y] # This is polynomials w coeffs in Q with variable x
    r = R(f)
    L = r.exponents()
    sumX = 0
    sumY = 0
    for k in range(len(L)):
        sumX += L[k][0]  # sum all x
        sumY += L[k][1]  # sum all y
    redPoint = (sumX/len(L), sumY/len(L))
#     print(redPoint)

    maxP = ((), 0)
    minP = ((), 99)
    for i in range(len(L)):
        dis = distance(redPoint, L[i])
        if dis > maxP[1]:
            maxP = (L[i], dis)
        if dis < minP[1]:
            minP = (L[i], dis)

    return (maxP, minP)


# f = -3*x^9*y+x^6*y^4+x^5*y^5+5*x^4*y^6-x^3*y^6+6*x^2*y^7-8*x^3*y^5-5*x^2*y^5
-4*y^3-2*x*y-2*x
# problem4(f)
problem4(-2*x^19*y - 2*x^14*y^6 - x^4*y^16 - 9*x^17*y^2 - x^15*y^4 + 7*x^10*y^
9 + 7*x^6*y^13
- x^12*y^6 - 2*x^9*y^9 + 9*x^8*y^10 + 9*x^5*y^13 - x^15*y^2 - x^6*y^11 - 12*x^
3*y^14 - 3*x^9*y^7
- 7*x^7*y^9 + x^5*y^11 + 2*x*y^14 + 5*x^13*y + x^10*y^3 + 2*x^6*y^7 + x^12 + x
^11*y - x^3*y^8
- 3*x^3*y^6 - 2*x^6*y^2 - x^5*y^3 + x*y^6 - x*y^3)
```

Out[9]: (((19, 1), 12.3103448275862), ((9, 7), 0.886548974633272))