K4bound

```
#dot product evaluation
def dp(x,d):
    a = d/2-1
    return (2/3*a^4*x^4+4*a^3*x^4+22/3*a^2*x^4-2*a^3*x^2+4*a*x^4-
6*a^2*x^2-4*a*x^2+1/2*a^2+1/2*a) / (2/3*a^4+2*a^3+11/6*a^2+1/2*a)
    #same as return expand(gegenbauer(4,d/2-1,x)/gegenbauer(4,d/2-
1,1))

def K4bound(n,k,l,m):
    #auxiliary variables
    aux_root = sqrt((l-m)^2+4*(k-m))
    aux_quot = (2*k+(n-1)*(l-m))/aux_root
    #positive eigenvalue with multiplicity
    r = (l-m+aux_root)/2
    f = (n-1-aux_quot)/2
    #negative eigenvalue with multiplicity
    s = (l-m-aux_root)/2
    g = (n-1+aux_quot)/2
    #inner products in R^g
    p = s/k
    q = -(s+1)/(n-k-1)

    #consider vectors V=(v_1+...) and E=(e_1+...), where all v_i
and e_i are unit vectors in R^g with the dot product computed via
the above function; if A=V^2, B=VE, C=E^2, by Cauchy-Schwartz
B^2<=AC

    #A=V^2
    A = n*(1+k*dp(p,g)+(n-k-1)*dp(q,g)) #with itself, edges, non-
edges

    #denominators of edges and squares
    normpp = 2+2*p
    normp = sqrt(normpp)

    #B=VE, fix vertex v, sum over different edges
    B = k*dp((1+p)/normp,g) #edge from v to N(v)
    B += k*l/2*dp(2*p/normp,g) #edges inside N(v)
    B += (n-k-1)*m*dp((p+q)/normp,g) #edges between N(v) and N'(v)
    B += (n-k-1)*(k-m)/2*dp(2*q/normp,g) #edges inside N'(v)
    B *= n #over all verticies

    #C=E^2, fix edge uv, sum over different edges
```

```
    #x=x_{uv} - number of edges in N(u)\cap N(v)
    #C=\sum_{uv}(C0+C1 x_{uv}) will be computing C0 and C1 below
    #\sum_{uv}x_{uv}=6 times number of K4
    C0 = dp(1,g) #with itself
    C1 = 0
    #vertices except u and v are split into
    #V_0 = N(u) \cap N(v) has l elements
    #V_1 = N(u) \cap N'(v) has k-l-1 elements
    #V_2 = N'(u) \cap N(v) has k-l-1 elements
    #V_3 = N'(u) \cap N'(v) has n-2k+l elements
    C0 += 2*l*dp((1+3*p)/normpp,g) #edges from u or v to V_0
    C0 += 2*(k-l-1)*dp((1+2*p+q)/normpp,g) #edges from u to V_1 or
from v to V_2
    C1 += dp(4*p/normpp,g) #edges in V_0
    #precomputing dot products, indexed by number of edges between
two edges that are multiplied
    dp3 = dp((3*p+q)/normpp,g)
    dp2 = dp((2*p+2*q)/normpp,g)
    dp1 = dp((p+3*q)/normpp,g)
    dp0 = dp(4*q/normpp,g)
    C0 += 2*l*(l-1)*dp3 #edges between V_0 and (V_1 or V_2)
Prop1(i)
    C1 -= 4*dp3
    C0 += l*(k-2*l)*dp2 #edges in V_1 or V_2 Prop1(ii)
    C1 += 2*dp2
    C0 += ((m-1)*(k-l-1)-l*(l-1))*dp2 #edges between V_1 and V_2
Prop1(iii)
    C1 += 2*dp2
    C0 += 2*((k-m)*(k-l-1)-l*(k-2*l))*dp1 #edges between V_3 and
(V_1 or V_2) Prop1(iv)
    C1 -= 4*dp1
    C0 += l*(k-2*l)*dp2 #edges between V_0 and V_3 Prop1(v)
    C1 += 2*dp2
    C0 += (k*(n-2*k+l)/2-(k-m)*(k-l-1)+l*(k-2*l)/2)*dp0 #edges in
V_3 Prop1(vi)
    C1 += dp0

    #C>=B^2/A
    if A==0 or C1<=0:
        return 0
    else:
        return ceil(((B^2/A-C0*n*k/2)/(6*C1)).n())
```

```
K4bound(76,30,8,14)
```

```
    39
```