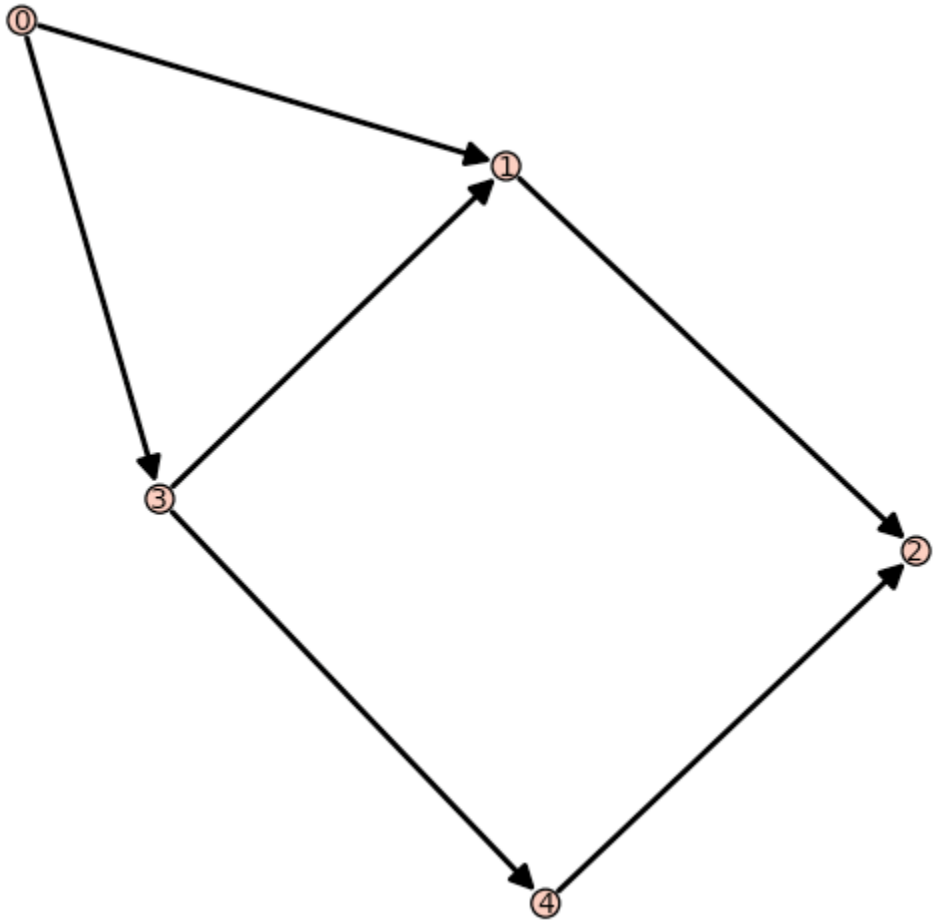
bott_residue tester

```
sys.path.append('/Users/bjones/sage_code/csm')
attach "/Users/bjones/sage_code/csm/bott_residue.py"

Z = Zresolution( [2,1], [1,1] )
FP = Z.fixed_points()
#for f in FP: print f
G=DiGraph(Z.graph)
G.plot(layout='spring', scaling_term=0.08, vertex_size=80)
```




```
a = ChernClass(1,'q',2,[1])
b = ChernClass(1,'q',2,[1,1])
M = ChernClassMonomial([a,b])
view(M);
```

$$s_{[1]}(\underline{Q}_2)s_{[1,1]}(\underline{Q}_2)$$

```
#
# The code below integrates the pull-back of c_[1,1](Q_2), a certain Chern
```

```
# class on the second natural quotient bundle over the Zelevinsky
# resolution (Z) of X_[2,1] defined above, times c(TZ).
#

c = ChernClass(1,'q',2,[1,1])

# X is a C^* character in C^4, this list doesn't matter as long as the integers
# are distinct
X = [0,1,2,3]

TTI = Z.tensor_term_iterator() # Computes terms in the total Chern class of the tangent bundle of Z
s = 0
for t in TTI:
  t.append(c)
  stemp = 0
  # sum the "Bott term" over the fixed points
  for fp in FP:
     stemp += bott_term_general(Z,fp,X,t)
  print ("t = %s"%t).ljust(72), "integral = %s" % stemp
  s += stemp
print "total sum =",s
```

```
t = 1 * ( s_[1](Q[1]) * s_[1, 1](Q[2]) * s_[1, 1](Q[2]) * )
integral = 0
t = 1 * ( s_[1](Q[1]) * s_[1](Q[2]) * s_[1, 1](Q[2]) * )
integral = 0
t = 1 * ( s_[1](Q[1]) * s_[1](S[2]) * s_[1](Q[2]) * s_[1, 1](Q[2]) *
)   integral = 0
t = 1 * ( s_[1](Q[1]) * s_[1, 1](Q[2]) * )
integral = 1
t = 2 * ( s_[1](Q[1]) * s_[1](S[2]) * s_[1, 1](Q[2]) * )
integral = 0
t = 1 * ( s_[1](Q[1]) * s_[2](S[2]) * s_[1, 1](Q[2]) * )
integral = 0
t = 1 * ( s_[1, 1](Q[2]) * s_[1, 1](Q[2]) * )
integral = 0
t = 1 * ( s_[1](Q[2]) * s_[1, 1](Q[2]) * )
integral = 1
t = 1 * ( s_[1](S[2]) * s_[1](Q[2]) * s_[1, 1](Q[2]) * )
integral = 0
t = 1 * ( s_[1, 1](Q[2]) * )
integral = 0
t = 2 * ( s_[1](S[2]) * s_[1, 1](Q[2]) * )
integral = 0
t = 1 * ( s_[2](S[2]) * s_[1, 1](Q[2]) * )
integral = 0
t = 1 * ( s_[1](S[1]) * s_[1, 1](Q[2]) * s_[1, 1](Q[2]) * )
integral = 0
t = 1 * ( s_[1](S[1]) * s_[1](Q[2]) * s_[1, 1](Q[2]) * )
integral = 0
t = 1 * ( s_[1](S[1]) * s_[1](S[2]) * s_[1](Q[2]) * s_[1, 1](Q[2]) *
)   integral = 0
t = 1 * ( s_[1](S[1]) * s_[1, 1](Q[2]) * )
```

```
integral = 1
t = 2 * ( s_[1](S[1]) * s_[1](S[2]) * s_[1, 1](Q[2]) * )
integral = 0
t = 1 * ( s_[1](S[1]) * s_[2](S[2]) * s_[1, 1](Q[2]) * )
integral = 0
total sum = 3
```