In [242]:
```
%matplotlib inline
```

In [243]:
```
import pylab as py
```

In [244]:
```
import numpy as np
```

In [245]:
```
from graph_tool.all import *
```

Creating a random Undirected Graph for evaluation

In [246]:
```
g=Graph(directed=False)
```

In [247]:
```
for i in range(0,5):
    g.add_vertex() ;
    print "added no",i;
"End"
```

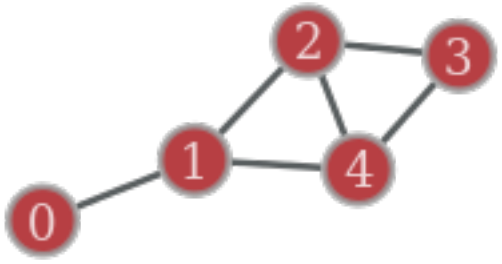
```
added no 0
added no 1
added no 2
added no 3
added no 4
```

Out[247]:
```
'End'
```

In [248]:
```
g.add_edge(g.vertex(0), g.vertex(1))
g.add_edge(g.vertex(1), g.vertex(2))
g.add_edge(g.vertex(2), g.vertex(3))
g.add_edge(g.vertex(4), g.vertex(3))
g.add_edge(g.vertex(1), g.vertex(4))
g.add_edge(g.vertex(2), g.vertex(4))
```

Out[248]:
```
<Edge object with source '2' and target '4' at 0x7f96f6f37f28>
```

In [249]:
```
graph_draw(g, vertex_text=g.vertex_index, vertex_font_size=18,output_size=(200, 200), output="two-nodes.png")
```

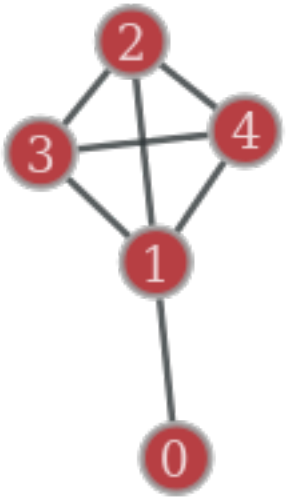


Out[249]:
```
<PropertyMap object with key type 'Vertex' and value type 'vector<double>', for Graph 0x7f96f6e6a390, at 0x7f96f6f9fd10>
```

In [250]:
```
g.add_edge(g.vertex(1), g.vertex(3))
```

Out[250]:
```
<Edge object with source '1' and target '3' at 0x7f96f6e5a3e0>
```

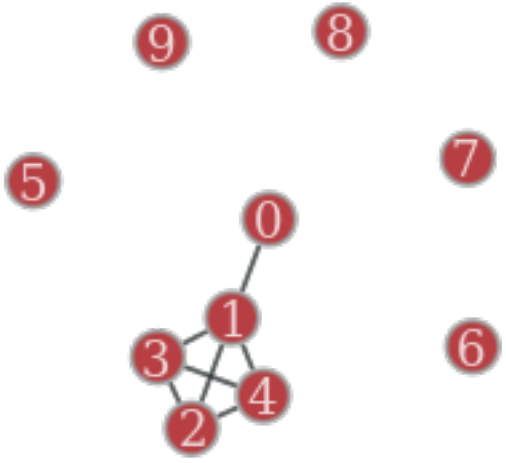
In [251]:
```
graph_draw(g, vertex_text=g.vertex_index, vertex_font_size=18,output_size=(200, 200), output="two-nodes.png")
```



Out[251]: <PropertyMap object with key type 'Vertex' and value type 'vector<double>', for Graph 0x7f96f6e6a390, at 0x7f96f6f9f290>

In [252]:
```
g.add_vertex(5);
```

In [253]:
```
graph_draw(g, vertex_text=g.vertex_index, vertex_font_size=18,output_size=(200, 200), output="two-nodes.png")
```



Out[253]:
```
<PropertyMap object with key type 'Vertex' and value type 'vector<double>', f
or Graph 0x7f96f6e6a390, at 0x7f96f6f9f250>
```

In [254]:
```
g.add_edge(g.vertex(7),g.vertex(4));
```

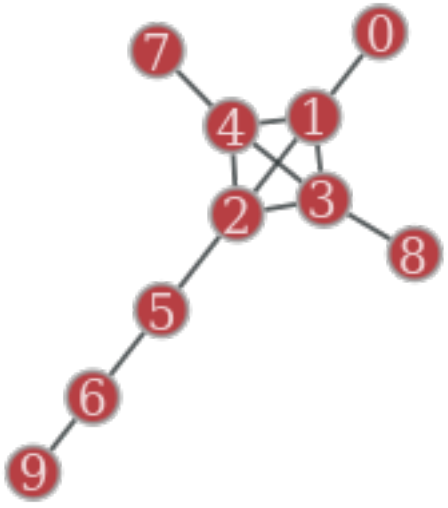
In [255]:
```
g.add_edge(g.vertex(8),g.vertex(3));
```

In [256]:
```
g.add_edge(g.vertex(2),g.vertex(5));
```

In [257]:
```
g.add_edge(g.vertex(6),g.vertex(9));
```

In [258]:
```
g.add_edge(g.vertex(5),g.vertex(6));
```

In [259]:
```
graph_draw(g, vertex_text=g.vertex_index, vertex_font_size=18,output_size=(20
0, 200), output="two-nodes.png")
```



Out[259]:
```
<PropertyMap object with key type 'Vertex' and value type 'vector<double>', f
or Graph 0x7f96f6e6a390, at 0x7f96f6f9f490>
```

Subroutine used by maxdsubgraph for modification of flow capacities

In [260]:
```
def modifycap(g,s,t,go,gn,cap):

    for e in g.edges():
        if  e.target()==t:
            cap[e]=cap[e]-2*go+2*gn
        "end"
    "end for"


"end func"
```

Out[260]:
```
'end func'
```

Subroutine which filters off unwanted nodes using the Maxflow min cut theoram

In [266]:

```
def maxdsubgraph(g,s,t,m,n,cap):
    go=0.0;
    l=0.0;
    u=m;
    b=1.0/(n*(n-1));
    res=-123;

    print "Inside MaxdSubgraph";


    for e in g.edges():
        print cap[e],e.source(),e.target();
    "end for"




    while (u-l)>b :

        print "value of l",l,"value of u",u,"value of b",b;

        gn=(u+l)/2;

        modifycap(g,s,t,go,gn,cap);

        print "yeah done with while";

        #"""


        #get cut
        #s=g.vertex(m);
```

```
        #t=g.vertex(m+1);


        res23 = push_relabel_max_flow(g, s, t, cap);

        #"""
        part = min_st_cut(g, s, cap, res23);
        #

        g.set_vertex_filter(part, inverted=False);




        if g.num_vertices()==1:
            print "u reduced";
            u=gn;

        else :
            print "l increased";
            l=gn;
            res=part;
        "end if"
        go=gn;
        g.clear_filters();
        #"""
    "end while"
    if(res!=-123):
        g.set_vertex_filter(res,inverted=False);
    "end if"


"end func"
```

```
Out[266]: 'end func'
```

Worker Function : Sets up thr priliminary G graph with dual edges and capacity 1

In [262]:

```
def dsg(g,m,n):

    print "m ",m,"n",n,"\n\n\n";

    G=Graph();

    G.add_vertex(n);

    for e in g.edges():
        G.add_edge(e.source(),e.target());
        G.add_edge(e.target(),e.source());
    "End for"

    s=G.add_vertex();
    t=G.add_vertex();




    for i in range(0,n):
        G.add_edge(s,G.vertex(i));
        G.add_edge(G.vertex(i),t);
    "End For"


    cap = G.new_edge_property("float")




    for e in G.edges():
        print "Source",e.source(),"target",e.target();
```

```
        if e.source()==s:
 #           print "found source";
            cap[e]=m;
        elif e.target()==t:
 #           print "found target";
            cap[e]=m-(e.source().in_degree()-1);
        else :
#           print "else mei";
            cap[e]=1;
        "end if"
    "end for"



    for e in G.edges():
        print cap[e],e.source(),e.target();
    "end for"

    maxdsubgraph(G,s,t,m,n,cap);


    cap2=g.new_vertex_property("bool");


    for v in g.vertices():
        cap2[v]=False;
    "end"

    for v in G.vertices():

        if v==s:
            "blah";
```

```
        else :

            cap2[v]=True;
        "end if"

    "end for"

    #res = boykov_kolmogorov_max_flow(G, s, t, cap);
    #part = min_st_cut(G, s, cap, res);
    #G.set_vertex_filter(part,inverted=False);

    g.set_vertex_filter(cap2,inverted=False);

    print "\n\n\n\nDensest Subgraph is ::";

    graph_draw(g, vertex_text=g.vertex_index, vertex_font_size=20,output_size
=(500, 500), output="two-nodes.png");


    print "Density is equal to",(g.num_edges()*1.0)/g.num_vertices();


"end def"
```

Out[262]: 'end def'

Densest Subgraph Function used on the graph created

In [263]:
```
dsg(g,g.num_edges(),g.num_vertices());
```

```
m  12 n 10


Source 0 target 1
Source 0 target 11
Source 1 target 0
Source 1 target 2
Source 1 target 4
Source 1 target 3
Source 1 target 11
Source 2 target 1
Source 2 target 3
Source 2 target 4
Source 2 target 5
Source 2 target 11
Source 3 target 1
Source 3 target 2
Source 3 target 4
Source 3 target 8
Source 3 target 11
Source 4 target 1
Source 4 target 2
Source 4 target 3
Source 4 target 7
Source 4 target 11
Source 5 target 2
Source 5 target 6
Source 5 target 11
Source 6 target 5
Source 6 target 9
Source 6 target 11
Source 7 target 4
```

```
Source 7 target 11
Source 8 target 3
Source 8 target 11
Source 9 target 6
Source 9 target 11
Source 10 target 0
Source 10 target 1
Source 10 target 2
Source 10 target 3
Source 10 target 4
Source 10 target 5
Source 10 target 6
Source 10 target 7
Source 10 target 8
Source 10 target 9
1.0 0 1
11.0 0 11
1.0 1 0
1.0 1 2
1.0 1 4
1.0 1 3
8.0 1 11
1.0 2 1
1.0 2 3
1.0 2 4
1.0 2 5
8.0 2 11
1.0 3 1
1.0 3 2
1.0 3 4
1.0 3 8
8.0 3 11
1.0 4 1
```

```
1.0 4 2
1.0 4 3
1.0 4 7
8.0 4 11
1.0 5 2
1.0 5 6
10.0 5 11
1.0 6 5
1.0 6 9
10.0 6 11
1.0 7 4
11.0 7 11
1.0 8 3
11.0 8 11
1.0 9 6
11.0 9 11
12.0 10 0
12.0 10 1
12.0 10 2
12.0 10 3
12.0 10 4
12.0 10 5
12.0 10 6
12.0 10 7
12.0 10 8
12.0 10 9
Inside MaxdSubgraph
1.0 0 1
11.0 0 11
1.0 1 0
1.0 1 2
1.0 1 4
1.0 1 3
```

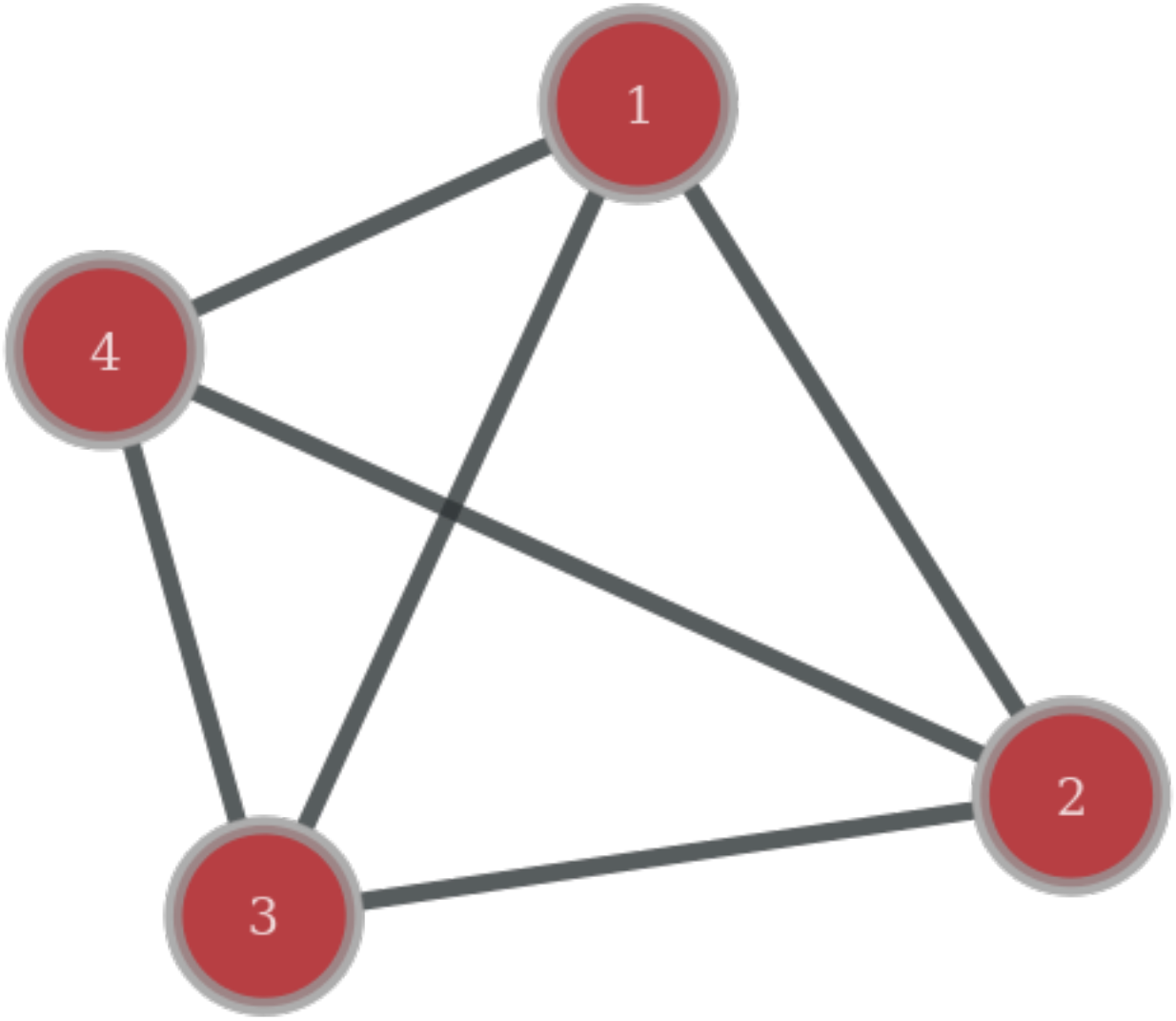
```
8.0 1 11
1.0 2 1
1.0 2 3
1.0 2 4
1.0 2 5
8.0 2 11
1.0 3 1
1.0 3 2
1.0 3 4
1.0 3 8
8.0 3 11
1.0 4 1
1.0 4 2
1.0 4 3
1.0 4 7
8.0 4 11
1.0 5 2
1.0 5 6
10.0 5 11
1.0 6 5
1.0 6 9
10.0 6 11
1.0 7 4
11.0 7 11
1.0 8 3
11.0 8 11
1.0 9 6
11.0 9 11
12.0 10 0
12.0 10 1
12.0 10 2
12.0 10 3
12.0 10 4
```

```
12.0 10 5
12.0 10 6
12.0 10 7
12.0 10 8
12.0 10 9
value of l 0.0 value of u 12 value of b 0.0111111111111
yeah done with while
u reduced
value of l 0.0 value of u 6.0 value of b 0.0111111111111
yeah done with while
u reduced
value of l 0.0 value of u 3.0 value of b 0.0111111111111
yeah done with while
u reduced
value of l 0.0 value of u 1.5 value of b 0.0111111111111
yeah done with while
l increased
value of l 0.75 value of u 1.5 value of b 0.0111111111111
yeah done with while
l increased
value of l 1.125 value of u 1.5 value of b 0.0111111111111
yeah done with while
l increased
value of l 1.3125 value of u 1.5 value of b 0.0111111111111
yeah done with while
l increased
value of l 1.40625 value of u 1.5 value of b 0.0111111111111
yeah done with while
l increased
value of l 1.453125 value of u 1.5 value of b 0.0111111111111
yeah done with while
l increased
value of l 1.4765625 value of u 1.5 value of b 0.0111111111111
```

```
yeah done with while
l increased
value of l 1.48828125 value of u 1.5 value of b 0.0111111111111
yeah done with while
l increased



Densest Subgraph is ::
```



Density is equal to 1.5

In [263]:

In [122]:

In []: