n-Cube

First we define the addition of two vectors $u, v \in \mathbb{Z}_2^n$

```
def dist(u,v):
    """
    Addition of two vectors in `Z_2^n`.

    EXAMPLES::

        sage: u=(1,0,1,1,1,0)
        sage: v=(0,0,1,1,0,0)
        sage: dist(u,v)
        2
    """
    h = [(u[i]+v[i])%2 for i in range(len(u))]
    return sum(h)
```

The distance function measures in how many slots two vectors in $\mathbb{Z}_2^n$ differ:

```
u=(1,0,1,1,1,0)
v=(0,0,1,1,0,0)
dist(u,v)
```

```
2
```

Now we are going to define the n-cube as the graph with vertices in $\mathbb{Z}_2^n$ and edges between vertex u and vertex v if they differ in one slot, that is, the distance function is 1.

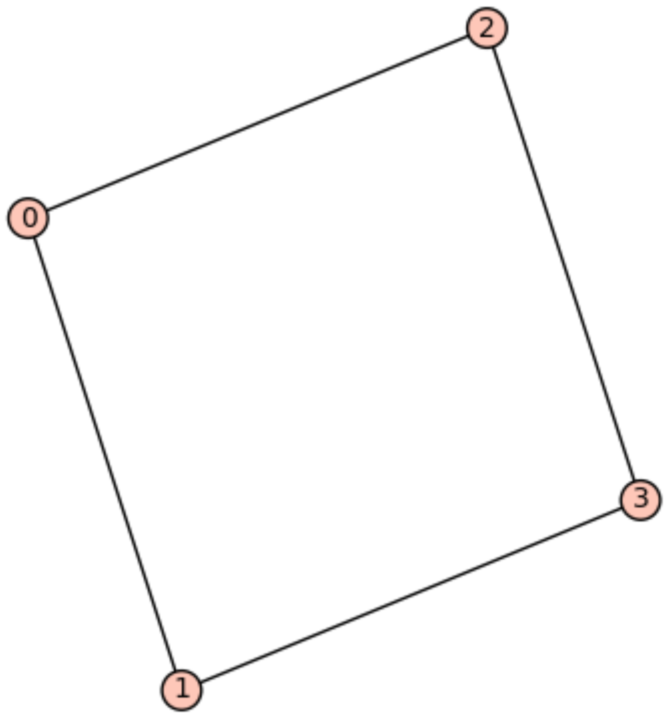
```
def cube(n):
    G = Graph(2**n)
    vertices = Tuples([0,1],n)
    for i in range(2**n):
        for j in range(2**n):
            if dist(vertices[i],vertices[j]) == 1:
                G.add_edge(i,j)
    return G
```

```
cube(4)
```

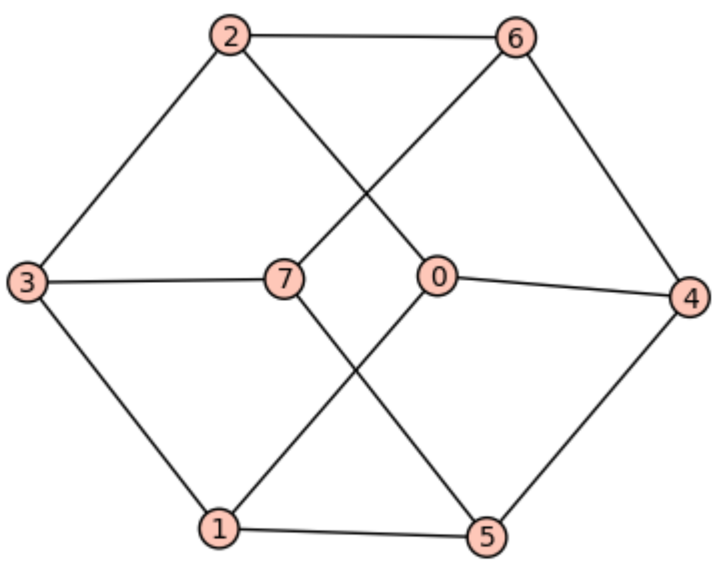
```
Graph on 16 vertices
```

We can now look at the 2, 3, and 4 dimensional cube:

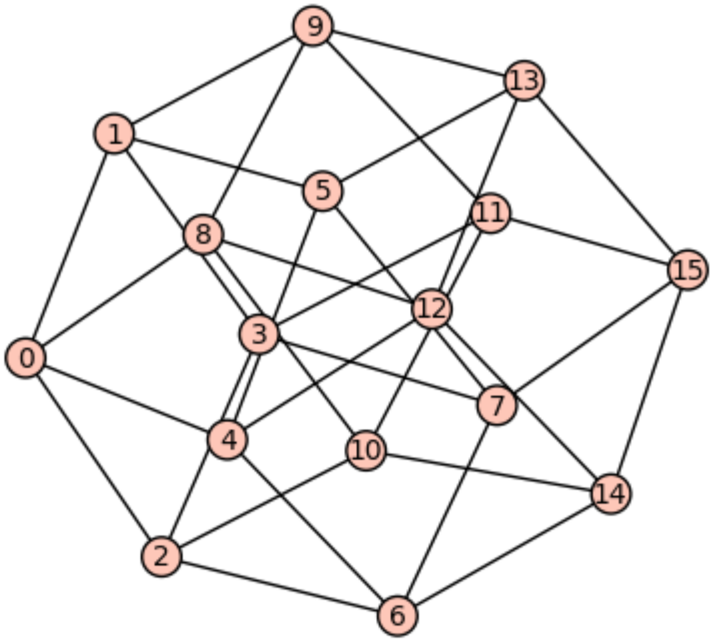
```
show(cube(2))
```




```
show(cube(3))
```




```
show(cube(4))
```



Now we can experiment and check Corollary 2.4 in Stanley's book:

```
G = cube(2)
G.adjacency_matrix().eigenvalues()
```

```
[2, -2, 0, 0]
```

```
G = cube(3)
G.adjacency_matrix().eigenvalues()
```

```
[3, -3, 1, 1, 1, -1, -1, -1]
```

```
G = cube(4)
G.adjacency_matrix().eigenvalues()
```

```
[4, -4, 2, 2, 2, 2, -2, -2, -2, -2, 0, 0, 0, 0, 0, 0]
```

It is easy now to slightly vary this problem and change the edge set by connecting vertices u and v if their distance is 2 (see Problem 3 on Homework 1):

```
def cube_2(n):
    G = Graph(2**n)
    vertices = Tuples([0,1],n)
    for i in range(2**n):
        for j in range(2**n):
            if dist(vertices[i],vertices[j]) == 2:
                G.add_edge(i,j)
    return G
```

```
G = cube_2(2);
G.adjacency_matrix().eigenvalues()
```

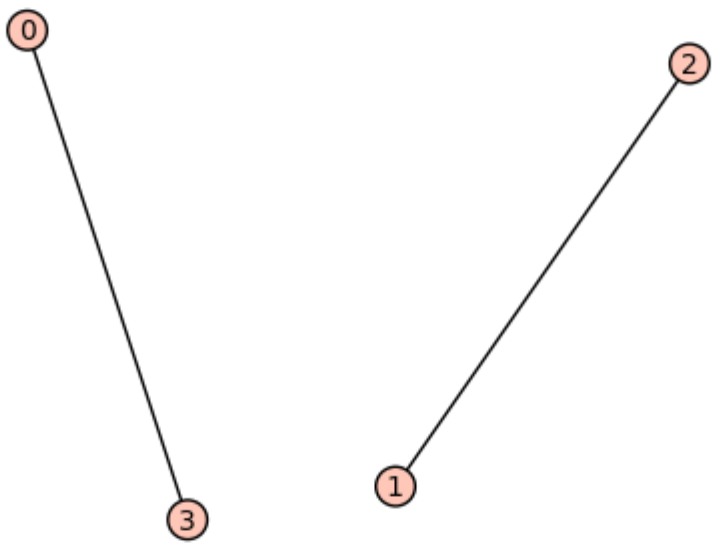
```
[1, 1, -1, -1]
```

```
G = cube_2(4);
G.adjacency_matrix().eigenvalues()
```

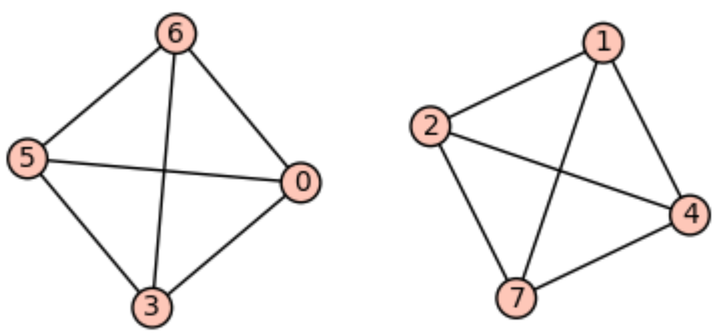
```
[6, 6, -2, -2, -2, -2, -2, -2, 0, 0, 0, 0, 0, 0, 0, 0]
```

Note that the graph is in fact disconnected:

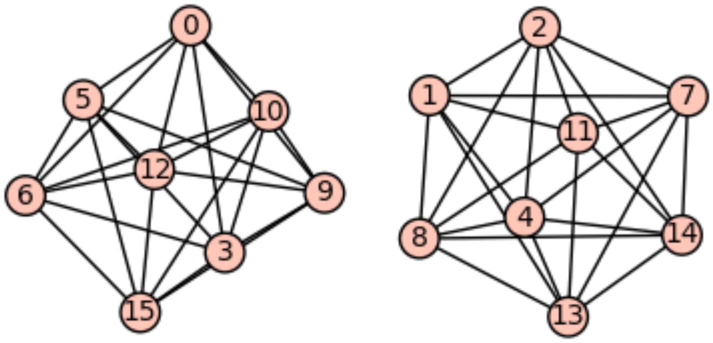
```
show(cube_2(2))
```



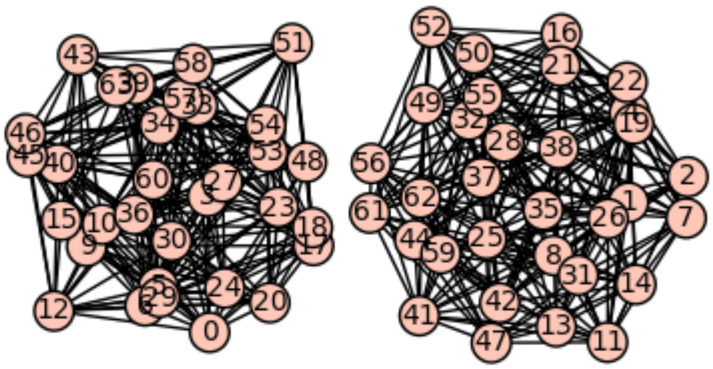

```
show(cube_2(3))
```




```
show(cube_2(4))
```


0
5
10
12
6
9
3
15
2
1
7
11
4
8
14
13

```
show(cube_2(6))
```


43
58
51
34
54
46
40
53
48
60
27
23
18
17
15
10
36
30
12
24
20
0
52
16
50
21
22
49
55
32
19
28
38
56
2
37
61
62
35
26
7
44
59
25
8
31
14
41
42
13
47
11