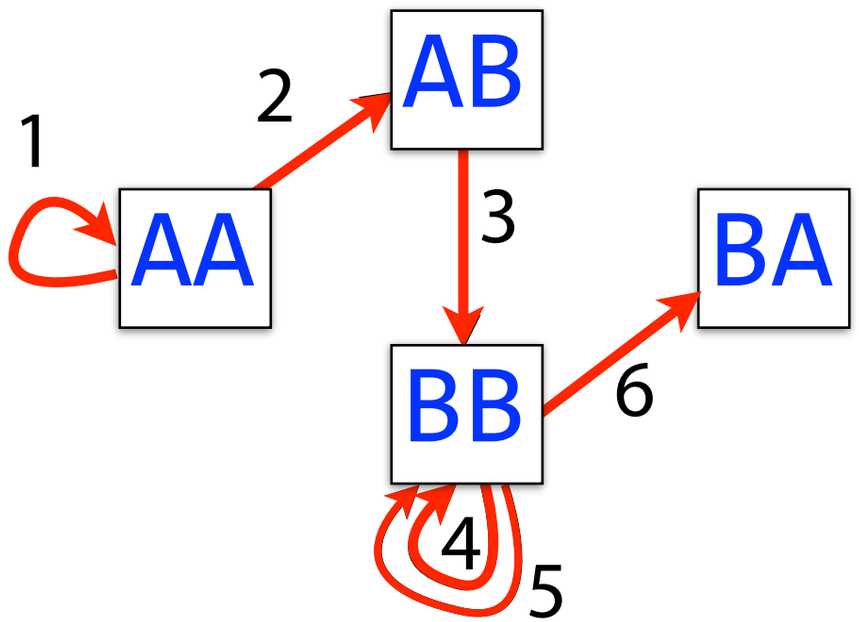


AAABBBBA

```
a_long_long_time
a_long long_t
 _long_ ong_ti
  long_l ng_tim
  long_l  g_time
   ong_lo
   ong_lo
    ng_lon
    ng_lon
     g_long
  g_long
    _long_
    _long_
```

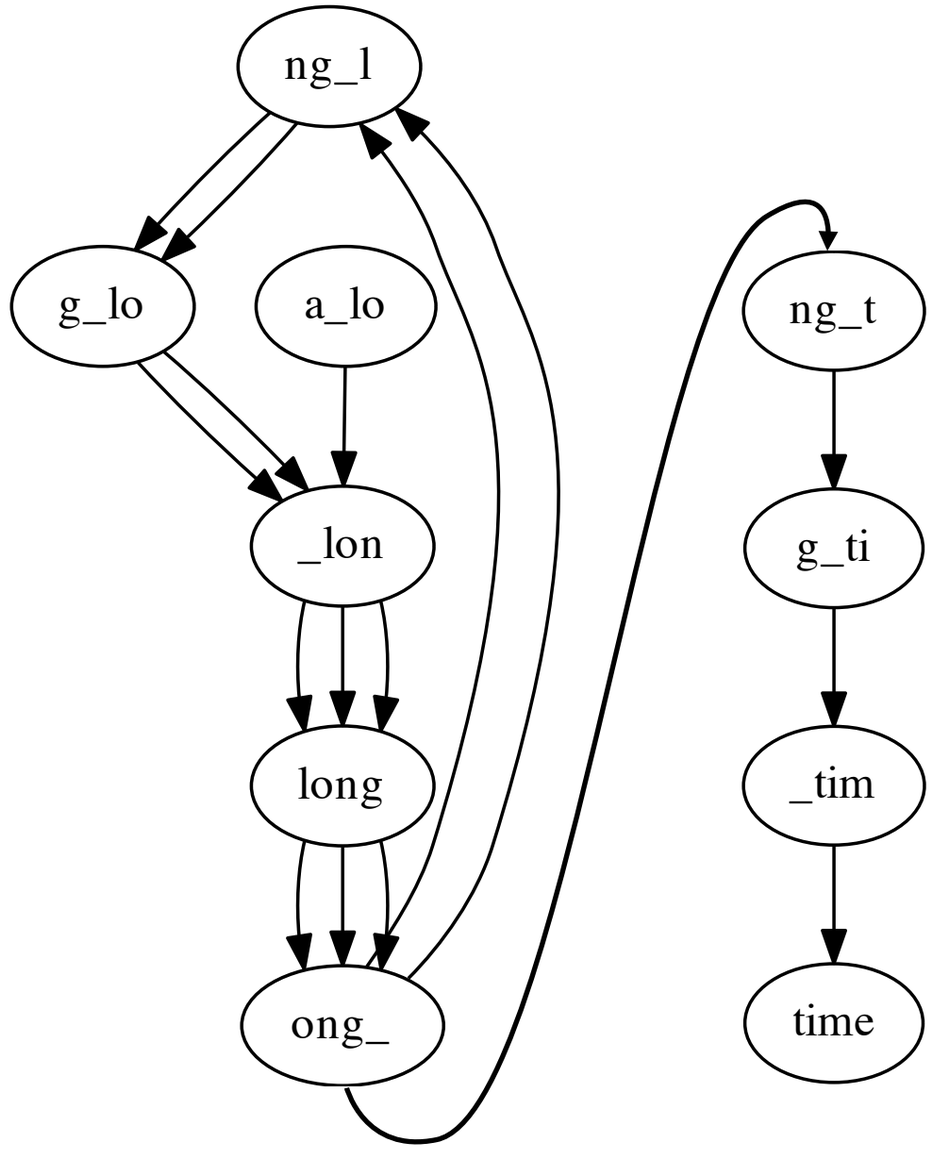
```
a_long_long_long_time
a_long long_l ng_tim
 _long_ ong_lo g_time
  long_l ng_lon
   ong_lo g_long
    ng_lon _long_
     g_long long_t
      _long_ ong_ti
```

```
a_long_long_long_long_long_time
a_long long_l  g_long   ng_tim
 _long_   ng_lon  long_l  g_time
   ong_lo  _long_   ng_lon
     g_long  ong_lo  _long_
                    long_t
                     ong_ti
```

De Bruijn graph (*k*=5) for:

a_long_long_long_time

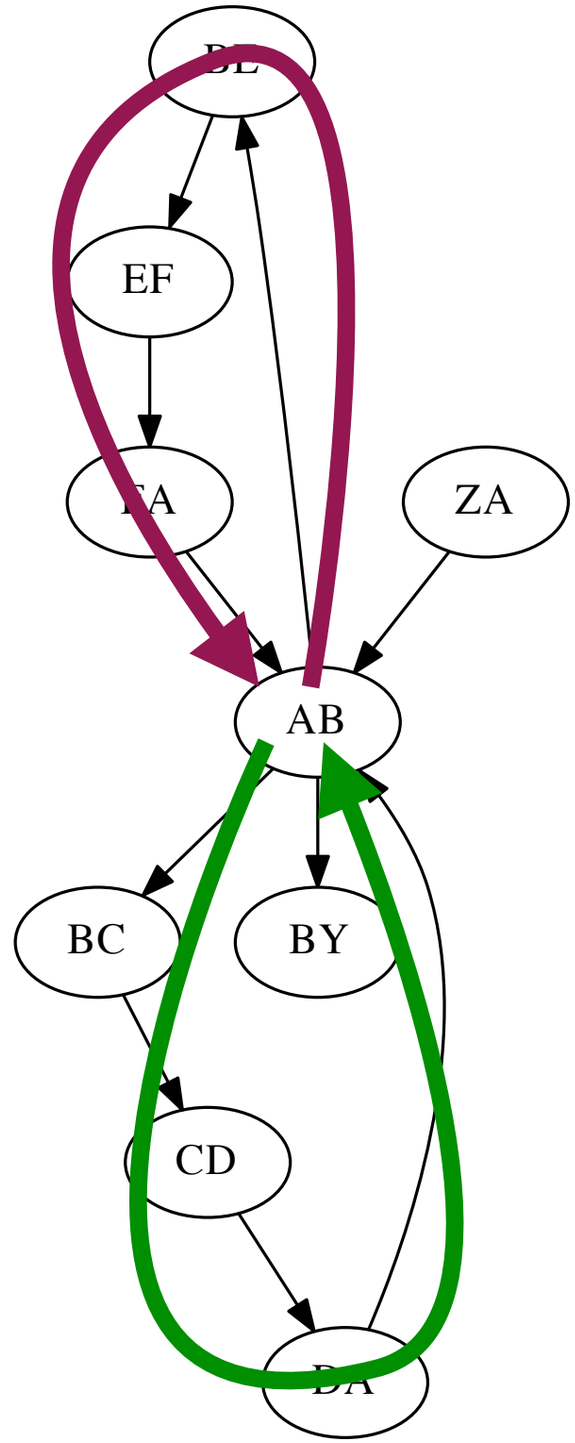
Eulerian walk gives original genome!



Right: graph for **ZABCDABEFABY**, $k = 3$

ZA → **AB** → **BE** → **EF** → **FA** → **AB** → **BC** → **CD** → **DA** → **AB** → **BY**

ZA → **AB** → **BC** → **CD** → **DA** → **AB** → **BE** → **EF** → **FA** → **AB** → **BY**



$k = 4$:

```
>>> st = "to_every_thing_turn_turn_turn_there_is_a_season"
>>> G = DeBruijnGraph([st], 4)
>>> path = G.eulerianWalk()
>>> superstring = path[0] + ''.join(map(lambda x: x[-1], path[1:]))
>>> print (superstring)
to_every_thing_turn_turn_turn_there_is_a_season
```

$k = 3$:

```
>>> st = "to_every_thing_turn_turn_turn_there_is_a_season"
>>> G = DeBruijnGraph([st], 3)
>>> path = G.eulerianWalk()
>>> superstring = path[0] + ''.join(map(lambda x: x[-1], path[1:]))
>>> print (superstring)
to_every_turn_turn_thing_turn_there_is_a_season
```

$k = 8$

Genome: `a_long_long_long_time`

Reads: `a_long_long_long`, `ng_long_l`, `g_long_time`

k-mers:

```
a_long_l          ng_long_   g_long_t
 _long_lo          g_long_l   _long_ti
  long_lon                    long_tim
   ong_long                    ong_time
    ng_long_
     g_long_l
      _long_lo
       long_lon
        ong_long
```