In [16]:
```
def dfa_accepts(start_state, delta, accept_states, string):
    """
    Return True if a DFA specified by the transition function `delta`,
    startingin state `start_state` and with accepting states
    `accept_states`, accepts the input `string`
    """
    cur_state = start_state
    for char in string:
        cur_state = delta[(cur_state,char)]
    return (cur_state in accept_states)
```

In [17]:
```
zero_parity_delta = { ('even','0') : 'odd', ('even','1') : 'even',
                ('odd', '0') : 'even', ('odd', '1') : 'odd'}
```

In [18]:
```
dfa_accepts('even', zero_parity_delta, { 'even' }, "00")
```

Out[18]: True

In [19]:
```
dfa_accepts('even', zero_parity_delta, { 'even' }, "000")
```

Out[19]: False

In [20]:
```
dfa_accepts('even', zero_parity_delta, { 'even' }, "011011010")
```

Out[20]: True

In [23]:
```
import time
import random
input_str = ''.join(["01"[random.randrange(2)] for _ in range(30000)])
start_time = time.time()
for _ in range(1000):
    dfa_accepts('even', zero_parity_delta, { 'even' }, input_str)
end_time = time.time()
print(f"Time taken {end_time-start_time:.3f}s")
```

```
Time taken 2.141s
```

In [24]:
```
def hanoi(n, s, t, x):
    """ Moves `n` disks from pole `s` to pole `t`, leaving `x` empty
    """
    if n > 0:
        hanoi(n-1, s, x, t)
        print(f"moving disk {n} from {s} to {t}")
        hanoi(n-1, x, t, s)
    else:
        pass
```

In [25]:
```
hanoi(5, 1, 2, 3)
```

```
moving disk 1 from 1 to 2
moving disk 2 from 1 to 3
moving disk 1 from 2 to 3
moving disk 3 from 1 to 2
moving disk 1 from 3 to 1
moving disk 2 from 3 to 2
moving disk 1 from 1 to 2
moving disk 4 from 1 to 3
moving disk 1 from 2 to 3
moving disk 2 from 2 to 1
moving disk 1 from 3 to 1
moving disk 3 from 2 to 3
moving disk 1 from 1 to 2
moving disk 2 from 1 to 3
moving disk 1 from 2 to 3
moving disk 5 from 1 to 2
moving disk 1 from 3 to 1
moving disk 2 from 3 to 2
moving disk 1 from 1 to 2
moving disk 3 from 3 to 1
moving disk 1 from 2 to 3
moving disk 2 from 2 to 1
moving disk 1 from 3 to 1
moving disk 4 from 3 to 2
moving disk 1 from 1 to 2
moving disk 2 from 1 to 3
moving disk 1 from 2 to 3
moving disk 3 from 1 to 2
moving disk 1 from 3 to 1
moving disk 2 from 3 to 2
moving disk 1 from 1 to 2
```