In [7]:
```
def queen_safe(row, qsofar):
    """ returns True if it is safe to place another
    queen in `row` in column `len(qsofar)`, given
    existing placement of queens in `qsofar` in the first
    `len(qsofar)` columns """
    col = len(qsofar)
    for (pcol,prow) in enumerate(qsofar):
        # can't place queen in same row
        if prow == row:
            return False
        # diagonal attack
        if abs(prow-row) == col-pcol:
            return False
    # no problems with any existing queens
    return True
```

In [11]:
```
def nqueens(n, qsofar=[]):
    col = len(qsofar)
    if col == n:
        return (True,0)
    safe_calls = 0
    for row in range(n):
        safe_calls += 1
        if queen_safe(row, qsofar):
            result, ncalls = nqueens(n, qsofar + [row])
            safe_calls += ncalls
            if result:
                return (True, safe_calls)
    return (False, safe_calls)
```

In [23]:
```
nq_complexity = []
for n in range(1,30):
    result = nqueens(n)
    print(n, result)
    nq_complexity.append(result[1])
```

```
1 (True, 1)
2 (False, 6)
3 (False, 18)
4 (True, 26)
5 (True, 15)
6 (True, 171)
7 (True, 42)
8 (True, 876)
9 (True, 333)
10 (True, 975)
11 (True, 517)
12 (True, 3066)
13 (True, 1365)
14 (True, 26495)
15 (True, 20280)
16 (True, 160712)
17 (True, 91222)
18 (True, 743229)
19 (True, 48184)
20 (True, 3992510)
21 (True, 179592)
22 (True, 38217905)
23 (True, 584591)
24 (True, 9878316)
25 (True, 1216775)
26 (True, 10339849)
27 (True, 12263400)

---------------------------------------------------------------------
-------
KeyboardInterrupt                         Traceback (most recent cal
l last)
<ipython-input-23-899f48fb0902> in <module>
      1 nq_complexity = []
      2 for n in range(1,30):
----> 3     result = nqueens(n)
      4     print(n, result)
      5     nq_complexity.append(result[1])

<ipython-input-11-0b6e55dc5c78> in nqueens(n, qsofar)
      7         safe_calls += 1
      8         if queen_safe(row, qsofar):
----> 9             result, ncalls = nqueens(n, qsofar + [row])
     10             safe_calls += ncalls
     11             if result:

<ipython-input-11-0b6e55dc5c78> in nqueens(n, qsofar)
      7         safe_calls += 1
      8         if queen_safe(row, qsofar):
----> 9             result, ncalls = nqueens(n, qsofar + [row])
```

```
     10             safe_calls += ncalls
     11             if result:

<ipython-input-11-0b6e55dc5c78> in nqueens(n, qsofar)
      7         safe_calls += 1
      8         if queen_safe(row, qsofar):
----> 9             result, ncalls = nqueens(n, qsofar + [row])
     10             safe_calls += ncalls
     11             if result:

<ipython-input-11-0b6e55dc5c78> in nqueens(n, qsofar)
      7         safe_calls += 1
      8         if queen_safe(row, qsofar):
----> 9             result, ncalls = nqueens(n, qsofar + [row])
     10             safe_calls += ncalls
     11             if result:

<ipython-input-11-0b6e55dc5c78> in nqueens(n, qsofar)
      7         safe_calls += 1
      8         if queen_safe(row, qsofar):
----> 9             result, ncalls = nqueens(n, qsofar + [row])
     10             safe_calls += ncalls
     11             if result:

<ipython-input-11-0b6e55dc5c78> in nqueens(n, qsofar)
      7         safe_calls += 1
      8         if queen_safe(row, qsofar):
----> 9             result, ncalls = nqueens(n, qsofar + [row])
     10             safe_calls += ncalls
     11             if result:

<ipython-input-11-0b6e55dc5c78> in nqueens(n, qsofar)
      7         safe_calls += 1
      8         if queen_safe(row, qsofar):
----> 9             result, ncalls = nqueens(n, qsofar + [row])
     10             safe_calls += ncalls
     11             if result:

<ipython-input-11-0b6e55dc5c78> in nqueens(n, qsofar)
      7         safe_calls += 1
      8         if queen_safe(row, qsofar):
----> 9             result, ncalls = nqueens(n, qsofar + [row])
     10             safe_calls += ncalls
     11             if result:

<ipython-input-11-0b6e55dc5c78> in nqueens(n, qsofar)
      7         safe_calls += 1
      8         if queen_safe(row, qsofar):
----> 9             result, ncalls = nqueens(n, qsofar + [row])
     10             safe_calls += ncalls
```

```
     11             if result:

<ipython-input-11-0b6e55dc5c78> in nqueens(n, qsofar)
      7         safe_calls += 1
      8         if queen_safe(row, qsofar):
----> 9             result, ncalls = nqueens(n, qsofar + [row])
     10             safe_calls += ncalls
     11             if result:

<ipython-input-11-0b6e55dc5c78> in nqueens(n, qsofar)
      7         safe_calls += 1
      8         if queen_safe(row, qsofar):
----> 9             result, ncalls = nqueens(n, qsofar + [row])
     10             safe_calls += ncalls
     11             if result:

<ipython-input-11-0b6e55dc5c78> in nqueens(n, qsofar)
      7         safe_calls += 1
      8         if queen_safe(row, qsofar):
----> 9             result, ncalls = nqueens(n, qsofar + [row])
     10             safe_calls += ncalls
     11             if result:

<ipython-input-11-0b6e55dc5c78> in nqueens(n, qsofar)
      7         safe_calls += 1
      8         if queen_safe(row, qsofar):
----> 9             result, ncalls = nqueens(n, qsofar + [row])
     10             safe_calls += ncalls
     11             if result:

<ipython-input-11-0b6e55dc5c78> in nqueens(n, qsofar)
      7         safe_calls += 1
      8         if queen_safe(row, qsofar):
----> 9             result, ncalls = nqueens(n, qsofar + [row])
     10             safe_calls += ncalls
     11             if result:

<ipython-input-11-0b6e55dc5c78> in nqueens(n, qsofar)
      7         safe_calls += 1
      8         if queen_safe(row, qsofar):
----> 9             result, ncalls = nqueens(n, qsofar + [row])
     10             safe_calls += ncalls
     11             if result:

<ipython-input-11-0b6e55dc5c78> in nqueens(n, qsofar)
      7         safe_calls += 1
      8         if queen_safe(row, qsofar):
----> 9             result, ncalls = nqueens(n, qsofar + [row])
     10             safe_calls += ncalls
     11             if result:
```

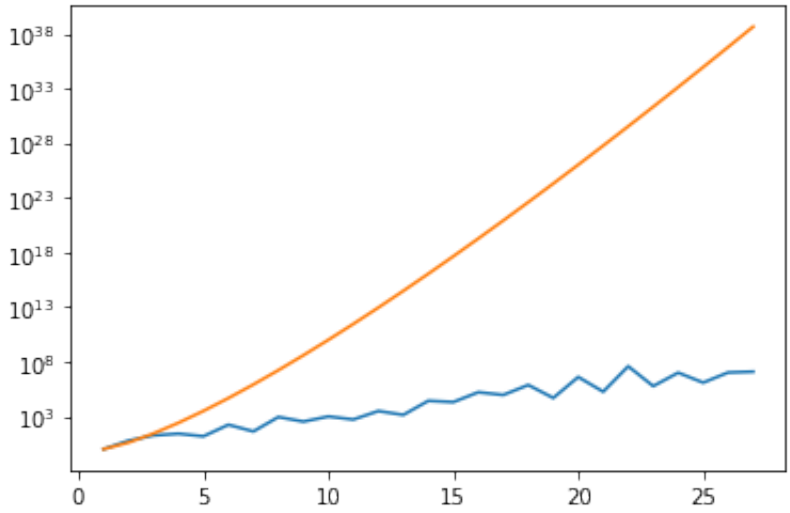
```
<ipython-input-11-0b6e55dc5c78> in nqueens(n, qsofar)
      7         safe_calls += 1
      8         if queen_safe(row, qsofar):
----> 9             result, ncalls = nqueens(n, qsofar + [row])
     10             safe_calls += ncalls
     11             if result:

<ipython-input-11-0b6e55dc5c78> in nqueens(n, qsofar)
      7         safe_calls += 1
      8         if queen_safe(row, qsofar):
----> 9             result, ncalls = nqueens(n, qsofar + [row])
     10             safe_calls += ncalls
     11             if result:

<ipython-input-11-0b6e55dc5c78> in nqueens(n, qsofar)
      7         safe_calls += 1
      8         if queen_safe(row, qsofar):
----> 9             result, ncalls = nqueens(n, qsofar + [row])
     10             safe_calls += ncalls
     11             if result:

<ipython-input-11-0b6e55dc5c78> in nqueens(n, qsofar)
      6     for row in range(n):
      7         safe_calls += 1
----> 8         if queen_safe(row, qsofar):
      9             result, ncalls = nqueens(n, qsofar + [row])
     10             safe_calls += ncalls

<ipython-input-7-85d5bce565ef> in queen_safe(row, qsofar)
     10             return False
     11         # diagonal attack
---> 12         if abs(prow-row) == col-pcol:
     13             return False
     14     # no problems with any existing queens

KeyboardInterrupt:
```

In [18]:

```
%matplotlib inline
import matplotlib.pyplot as plt
```

In [26]:
```
xaxis = list(range(1,len(nq_complexity)+1))
plt.yscale('log')
plt.plot(xaxis,nq_complexity)
plt.plot(xaxis, [n**n for n in xaxis])
```

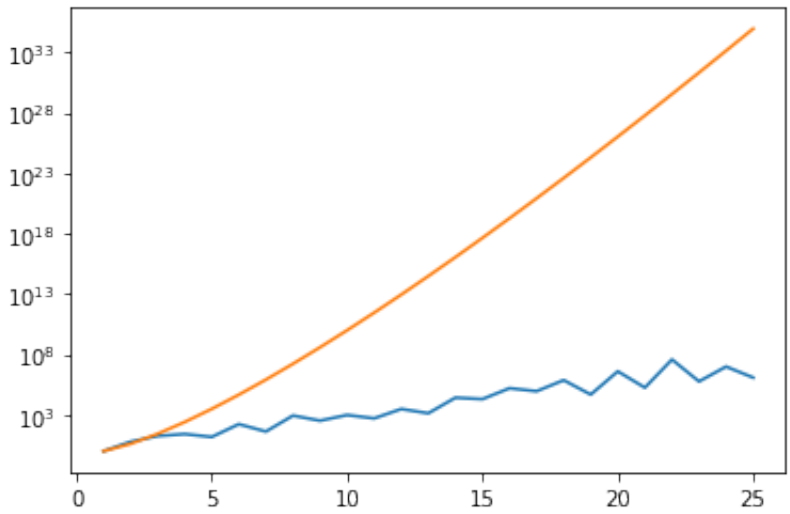
Out[26]: [<matplotlib.lines.Line2D at 0x121499a90>]



In [20]:
```
xaxis = list(range(1, len(nq_complexity)+1))
```

In [22]:
```
plt.yscale('log')
plt.plot(xaxis, nq_complexity)
plt.plot(xaxis, [n**n for n in xaxis])
```

Out[22]: [<matplotlib.lines.Line2D at 0x11c3abd90>]



In [53]:
```
words = {"this","is","a","hard","course", "aha", "i", "a", "ah", "ha",
"his", "sis", "our","ours","har"}
```

In [54]:
```
def segment(s, n=0):
    print(f"segment {s[n:]}")
    if n == len(s):
        return True
    for i in range(1, len(s)-n+1):
        if s[n:n+i] in words:
            if segment(s,n+i):
                return True
    return False
```

In [57]:
```
segment("thisisahardcourse")
```

```
segment thisisahardcourse
segment isahardcourse
segment sahardcourse
segment ahardcourse
segment hardcourse
segment rdcourse
segment dcourse
segment course
segment
```

Out[57]: True

In [58]:
```
segment("thisisahardcourseq")
```

```
segment thisisahardcourseq
segment isahardcourseq
segment sahardcourseq
segment ahardcourseq
segment hardcourseq
segment rdcourseq
segment dcourseq
segment courseq
segment q
segment ardcourseq
segment rdcourseq
segment rdcourseq
```

Out[58]: False