In [5]:

```
import time

problemSize = 10000000
print("%12s%16s" % ("Problem Size", "Seconds"))

for count in range(5):
    start = time.time()
    # The start of the algorithm
    work = 1
    for x in range(problemSize):
        work += 1
        work -= 1
    # The end of the algorithm
    elapsed = time.time() - start
    print("%12d%16.3f" % (problemSize, elapsed))
    problemSize *= 2
```

```
Problem Size         Seconds
    10000000           3.172
    20000000           6.275
    40000000          12.989
    80000000          25.231
   160000000          50.408
```

In [13]:

```
problemSize = 1000
print("%12s%16s" % ("Problem Size", "Seconds"))

for count in range(5):
    start = time.time()
    # The start of the algorithm
    work = 1
    for x in range(problemSize):
        work += 1
        work -= 1
    # The end of the algorithm
    elapsed = time.time() - start
    print("%12d%16.3f" % (problemSize, elapsed))
    problemSize *= 2
```

```
Problem Size         Seconds
        1000           0.000
        2000           0.002
        4000           0.001
        8000           0.003
       16000           0.004
```

In [15]:

```
problemSize = 1000
print("%12s%15s" % ("Problem Size", "Iterations"))

for _ in range(5):
    number = 0
    # The start of the algorithm
    work = 1
    for _ in range(problemSize):
        for _ in range(problemSize):
            number += 1
            work += 1
            work -= 1
    # The end of the algorithm
    print("%12d%15d" % (problemSize, number))
    problemSize *= 2
```

```
Problem Size     Iterations
        1000        1000000
        2000        4000000
        4000       16000000
        8000       64000000
       16000      256000000
```

In [16]:

```python
class Counter:
    def __init__(self):
        self.count = 0

    def increment(self):
        self.count += 1

    def __str__(self):
        return str(self.count)


def fib(n, counter):
    """Count the number of calls of the Fibonacci function."""
    counter.increment()
    if n < 3:
        return 1
    else:
        return fib(n - 1, counter) + fib(n - 2, counter)


problemSize = 2
print("%12s%15s" % ("Problem Size", "Calls"))

for _ in range(5):
    counter = Counter()
    # The start of the algorithm
    fib(problemSize, counter)
    # The end of the algorithm
    print("%12d%15s" % (problemSize, counter))
    problemSize *= 2
```

```
Problem Size          Calls
           2              1
           4              5
           8             41
          16           1973
          32        4356617
```

In [ ]: