????1

October 2, 2019

In [67]:
```
import numpy as np
import time
import sys

#配列の最大値を求める関数
def max(x):
    y=x[0]
    for i in x:
        if i>y:
            y=i
    return y
```

この関数を用いてある個数のデータの最大値を求めるのに、何秒かかるかを測定する。
ここでは 1000,2000,3000,.....,10000000 個の 10000 種類について比較検討する。

In [68]:
```
n=np.arange(1000,1000001,1000)
```

In [69]:
```
time_save=[]
for i in n:
    start=time.time()
    l=np.random.rand(i)
    print(max(l),"     ",end="")
    fin=time.time()
    time_save.append(fin-start)
```

これをもとにプロットしたところ、個数と時間の関係は、横軸個数、縦軸時間として以下のようになった

In [70]:
```
import matplotlib.pyplot as plt
plt.plot(n,time_save)
```

Out[70]:
```
[<matplotlib.lines.Line2D at 0x1166047f0>]
```

上のグラフから、計算時間は、おおよそ個数に比例して変化すると考えられる。つまり、このアルゴリズムの時間複雑度は、O(n) であると言える。

In [ ]:

In [ ]: