結果

最大値を求める関数を以下のようにpythonで実装した。

#max.py

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した。1000,2000,3000,.....,10000000個の10000種類のデータ配列について調べたところ、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。プログラムは以下のように実装した。

import matplotlib.pyplot as plt

time\_save=[]

for i in n:

start=time.time()

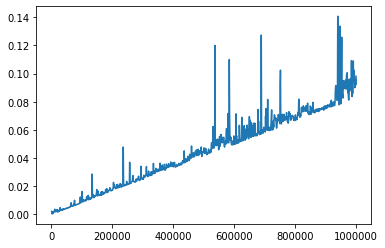
l=np.random.rand(i)

m=max(l)

fin=time.time()

time\_save.append(fin-start)

plt.plot(n,time\_save)



考察

上のグラフから、計算時間は、おおよそ個数に比例して上昇すると考えられる。

つまり、このアルゴリズムの時間複雑度は、帰納的にO(n)であると言える。

また、プログラムから演繹的にオーダーを求めてみると、計算時間は、

def max(x):

y=x[0] a秒

for i in x:

if i>y: b秒

y=I ｃ秒

return y ｄ秒

それぞれの処理にかかる時間をa,b,c,dとおくと、

となるので、になると考えられる。