**Eratosthen**

from math import ceil

def eratosthenus(n):

if n<2:

return []

r=[True]\*(n+1)

r[0]=r[1]=False

for i in range(2,int(n\*\*.5+1)):

if r[i]:

r[2\*i:n+1:i]=[False]\*ceil(((n+1)/i)-2)

return [i for i,j in enumerate(r) if j]

**Miller-Rabin test**

from random import randint

def miller\_rabin\_test(n,k=5):

if n<2:

return False

if n<4:

return True

r,s=0,n-1

while s%2==0:

r+=1

s//=2

for \_ in range(k):

a=randint(2,n-1)

x=pow(a,s,n)

if x==1 or x==n-1:

continue

for \_\_ in range(r-1):

x=pow(x,2,n)

if x==n-1:

break

else:

return False

return True

**determinant matrix**

def d(m):

if len(m)==1:

return m[0][0]

if len(m)==2:

return m[0][0]\*m[1][1]-m[0][1]\*m[1][0]

return sum(((-1)\*\*j)\*v\*d(f(m,0,j)) for j,v in enumerate(m[0]))

def f(m,x,y):

return [[v for j,v in enumerate(r) if j!=y] for i,r in enumerate(m) if i!=x]

**matrix product**

def matrix\_product(a,b):

if len(a[0])==len(b):

return [[sum(a[i][j]\*b[j][k] for j in range(len(b))) for k in range(len(b[0]))] for i in range(len(a))]

**max sum path in matrix**

def find\_sum(a):

n,m=len(a),len(a[0])

r=[[0]\*(m+1) for \_ in range(n+1)]

for i in range(1,n+1):

for j in range(1,m+1):

r[i][j]=max(r[i-1][j],r[i][j-1])+a[i-1][j-1]

return r[n][m]

**algoritm prima**

from math import inf

from gen\_graph import p\_graph

def get\_min(R,U):

rm=(inf,'Z','Z')

for v in U:

rr=min(R,key=lambda x: x[0] if (x[1]==v or x[2]==v) and (x[1] not in U or x[2] not in U) else inf)

if rm[0]>rr[0]:

rm=rr

return rm

def prim(a):

s=list(set(''.join([i[1]+i[2] for i in a])))

n=len(s)

a=[(inf,'Z','Z')]+a

u={s[0]}

r=[]

while len(u)<n:

m=get\_min(a,u)

if m[0]==inf:

break

r.append(m)

u.update([m[1],m[2]])

return r