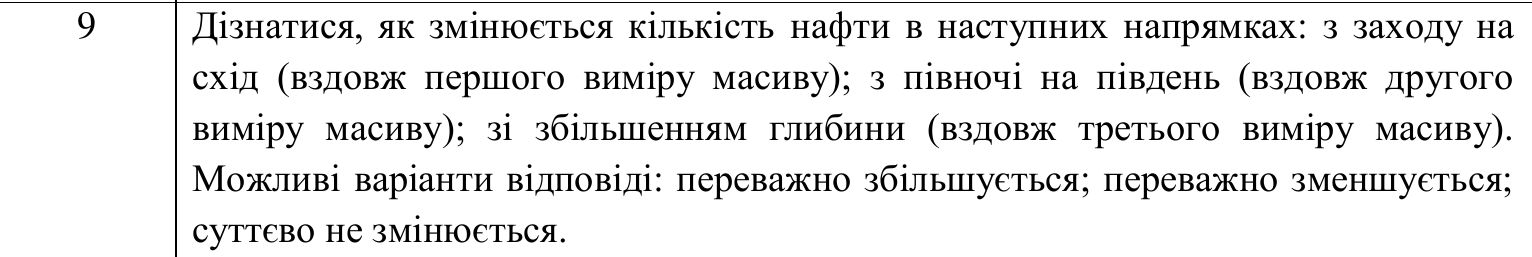
Варіант завдання



1. Алгоритм роботи програми
2. Даний алгоритм відтворює алгоритм попередньої роботи
3. функція генерації початкового тривимірного масиву модифікована

додаванням побудови розподілу кількості згенерованих мінералів

1. виведення результатів модифіковано виведенням графіків зміни кількості нафти в різних напрямках а також гістограмою кількості нафти в залежності від локації відносно перших двох масивів
2. Лістинг програми

import time, random

import numpy as np

import matplotlib.pyplot as plt

def oil\_checker(map\_,mod):

result = []

sum\_ = 0

if mod == 0:

for i in range(len(map\_)):

for j in range(len(map\_[i])):

for l in range(len(map\_[i][j])):

if map\_[i][j][l] == 4:

sum\_+=1

result.append(sum\_)

sum\_ = 0

return result

elif mod == 1:

for j in range(len(map\_[0])):

for i in range(len(map\_)):

for l in range(len(map\_[i][j])):

if map\_[i][j][l] == 4:

sum\_+=1

result.append(sum\_)

sum\_ = 0

return result

elif mod == 2:

for l in range(len(map\_[0][0])):

for i in range(len(map\_)):

for j in range(len(map\_[0])):

if map\_[i][j][l] == 4:

sum\_+=1

result.append(sum\_)

sum\_ = 0

return result

else:

print('Wrong mod')

return

def check\_feasibility(profit):

count\_inc = 0

count\_dec = 0

for i in range(len(profit)-1):

if profit[i+1] > profit[i]:

count\_inc+=1

elif profit[i+1] < profit[i]:

count\_dec+=1

try:

if (count\_inc-count\_dec)/(count\_inc+count\_dec)>0.1:

return 1

elif (count\_inc-count\_dec)/(count\_inc+count\_dec)<-0.1:

return -1

else:

return 0

except ZeroDivisionError:

return 0

def decoder(res):

if res==1:

return "amount of oil increase"

elif res==-1:

return "amount of oil decrease"

elif res==0:

return "amount of oil approximately stable"

else:

return "wrong function input"

def initial\_data(add\_oil=False):

m = int(input('m = '))

n = int(input('n = '))

k = int(input('k = '))

if any(i<3 for i in [m,n,k]):

print("m,n,k should be larger than 2")

initial\_data()

map\_ = [[[random.randint(0,7) for l in range(k)] for j in range(n)] for i in range(m)]

if add\_oil:

for i in range(int(m\*n\*k/2)):

map\_[random.randint(0,m-1)][random.randint(0,n-1)][random.randint(0,k-1)] = 4

plt.plot()

plt.hist([map\_[i][j][l] for i in range(m) for j in range(n) for l in range(k)],bins=8)

plt.xlabel('mineral')

plt.ylabel('Amount')

# plt.show()

return map\_

if \_\_name\_\_ == '\_\_main\_\_':

print("###########general###########")

map\_ = initial\_data()

time\_start = time.time()

along\_x = oil\_checker(map\_,0)

along\_y = oil\_checker(map\_,1)

along\_z = oil\_checker(map\_,2)

res\_x = check\_feasibility(along\_x)

res\_y = check\_feasibility(along\_y)

res\_z = check\_feasibility(along\_z)

fig, axs = plt.subplots(1,3)

fig.suptitle('Oil distribution along each direction')

axs[0].plot([i for i in range(len(along\_x))],along\_x)

axs[1].plot([i for i in range(len(along\_y))],along\_y)

axs[2].plot([i for i in range(len(along\_z))],along\_z)

axs[0].set\_xlabel('W-E')

axs[1].set\_xlabel('S-N')

axs[2].set\_xlabel('Depth')

axs[0].set\_xlabel('Amount')

axs[1].set\_xlabel('Amount')

axs[2].set\_xlabel('Amount')

# plt.show()

time\_finish = time.time()

# print(map\_)

# print("From E to W",along\_x)

# print("From S to N",along\_y)

# print("Into the depth",along\_z)

# print("From E to W",decoder(res\_x))

# print("From S to N",decoder(res\_y))

# print("Into the depth",decoder(res\_z))

general\_time = time\_finish-time\_start

print("t = ",general\_time)

print("###########numpy###########")

m = len(map\_)

n = len(map\_[0])

k = len(map\_[0][0])

map\_np = np.zeros((m,n,k))

for i in range(m):

for j in range(n):

for l in range(k):

map\_np[i][j][l] = map\_[i][j][l]

time\_start = time.time()

arr\_np = np.zeros((m, n, k)).astype(int)

arr\_np[(map\_np == 4)] = 1

along\_x = np.sum(np.sum(arr\_np,axis=2),axis=1)

along\_y = np.sum(np.sum(arr\_np.transpose(1,0,2),axis=2),axis=1)

along\_z = np.sum(np.sum(arr\_np.transpose(2,0,1),axis=2),axis=1)

res\_x = check\_feasibility(along\_x)

res\_y = check\_feasibility(along\_y)

res\_z = check\_feasibility(along\_z)

time\_finish = time.time()

fig = plt.figure()

ax = fig.add\_subplot(projection='3d')

x = np.array([])

y = np.array([])

depth\_columns = np.sum(arr\_np,axis=2)

for i in range(len(depth\_columns)):

for j in range(len(depth\_columns[0])):

for l in range(depth\_columns[i][j]):

x = np.append(x,i)

y = np.append(y,j)

hist, xedges, yedges = np.histogram2d(x, y, range=[[0, len(depth\_columns)], [0, len(depth\_columns[0])]])

xpos, ypos = np.meshgrid(xedges[:-1] + 0.25, yedges[:-1] + 0.25, indexing="ij")

xpos = xpos.ravel()

ypos = ypos.ravel()

zpos = 0

dx = dy = 0.5 \* np.ones\_like(zpos)

dz = hist.ravel()

ax.bar3d(xpos, ypos, zpos, dx, dy, dz, zsort='average')

plt.title('Amount of oil')

plt.xlabel('X')

plt.ylabel('Y')

plt.show()

#print(map\_)

# print("From E to W",along\_x)

# print("From S to N",along\_y)

# print("Into the depth",along\_z)

# print("From E to W",decoder(res\_x))

# print("From S to N",decoder(res\_y))

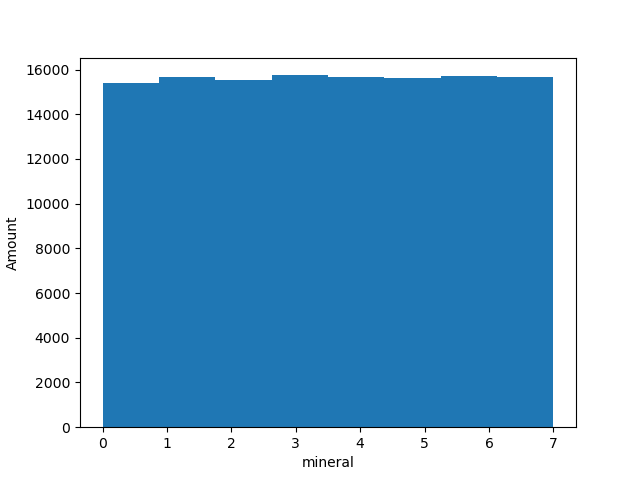
# print("Into the depth",decoder(res\_z))

print("t = ",time\_finish-time\_start)

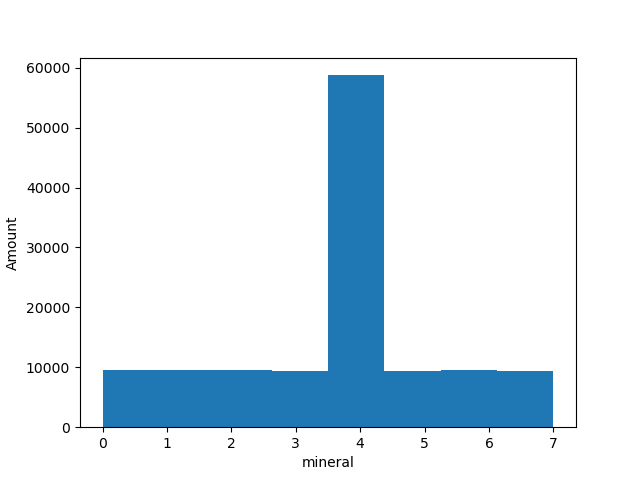
print("N = ",(general\_time/(time\_finish-time\_start)))

1. Результат роботи програми

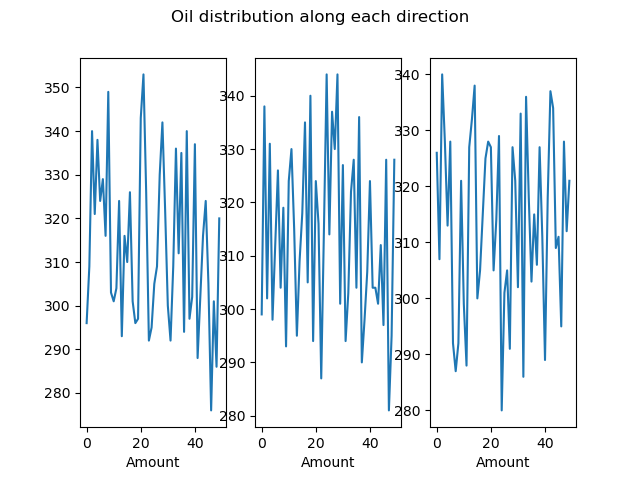
Гістограма розподілу згенерованих мінералів без збагачення нафтою:



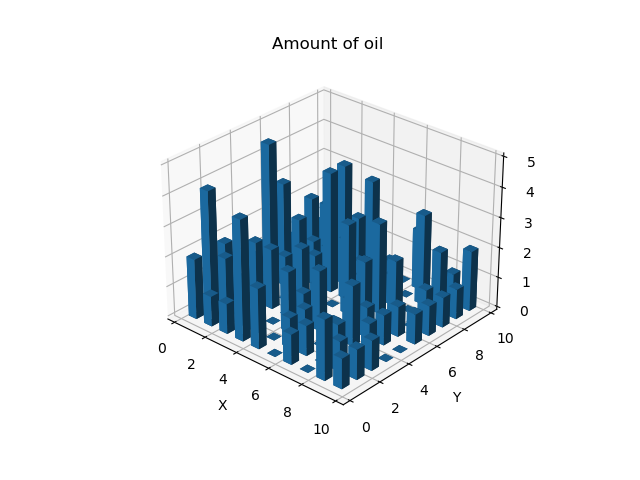
Гістограма розподілу згенерованих мінералів з збагаченням нафтою:



Графіки кількості розподілу кількості нафти у заданому напрямку:



Графік кількості нафти в залежності від локації відносно перших двох масивів:



1. Висновки

В даній лабораторній роботі я отримав практичні навички роботи з бібіліотекою matplotlib. Як рузультат мною була написана програма, що модернізує програмний код роботи 2 візуалізацію результатів роботи 2.