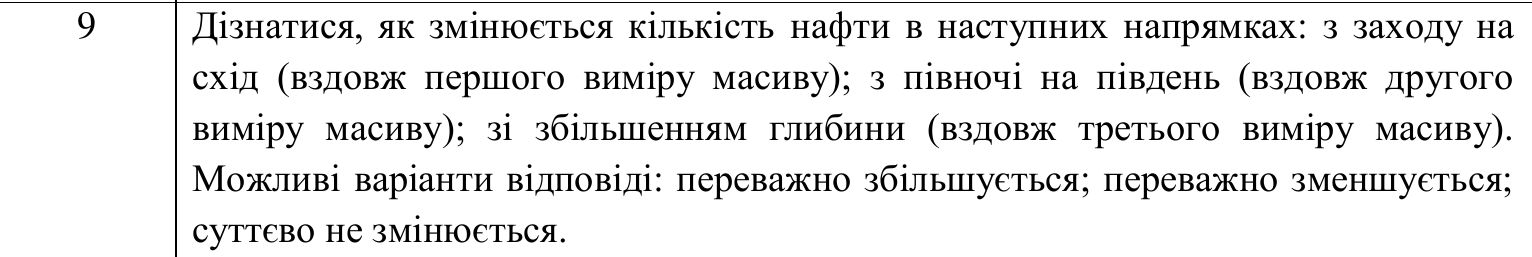
Варіант завдання



1. Алгоритм роботи програми
2. Проводиться алгоритм роботи 1
3. створюється заповнений нулями тривимірний масив, який має той же розмір, що і початковий після чого він заповнюється тими ж значеннями
4. фіксується час старт алгоритму
5. створюємо маску для створеного масиву
6. підсумовуємо значення вздовж необхідних осей(використовуємо метод transpose (для масиву розмірності n, якщо задано осі, їх порядок вказує, як осі переставлено) і маємо як результат масиви, що репрезентують кількість нафти у певному напрямку
7. перевіряємо масиви на предмет зростання кількості нафти
8. фіксуємо час кінця алгоритму
9. виводимо результати
10. Лістинг програми

import time, random

import numpy as np

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

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)

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()

# 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))

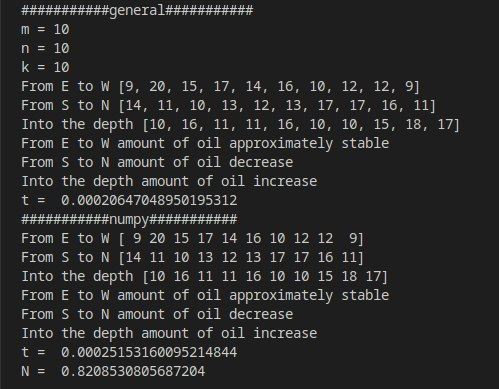
# print("Into the depth",decoder(res\_z))

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

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

1. Результат роботи програми

Для демонстраціїї роботи програми згенеровано земельну ділянку розмірами 10\*10\*10



З виводу програми видно:

кількість нафти з заходу на схід (вздовж першого виміру масиву m) збільшується;

кількість нафти з півночі на південь (вздовж другого виміру масиву n) переважно зменшується;

кількість нафти зі збільшенням глибини (вздовж третього виміру масиву k) переважно зменшується.

Висновок програми повність співпадає з реальними цифрами.

Також було проведено ряд дослідів з різними розмірами масиву:

|  |  |  |  |
| --- | --- | --- | --- |
| m\*n\*k | t1 | t2 | N |
| 165\*165\*165 | 1 | 0.03 | 35.8 |
| 350\*350\*350 | 10 | 0.22 | 46.7 |
| 495\*495\*495 | 30 | 0.60 | 46.0 |
| 625\*625\*625 | 60 | 2.27 | 27.3 |
| 780\*780\*780 | 120 | 9.91 | 12.9 |

1. Висновки

В даній лабораторній роботі я отримав практичні навички роботи з бібіліотекою numpy. Як рузультат мною була написана програма, що модернізує програмний код роботи 1 додаючи аналогічну обробку тривимірних масивів за допомогою numpy. В результаті роботи програми було показано, що результити роботи обох методів співпадають, проте швидкість роботи програми за допомогою numpy у порівнянні зі швидкістю роботи зі звичайними масивами є суттєво більшою