**Seminar 14 – AP**

Complexitati:

T(n) – numar de pasi

O – marginea superioara a numarului de pasi; T(n) <= O

Θ – exact numarul de pasi; T(n) = Θ

Ω - marginea inferioara a numarului de pasi; T(n) >= Ω

Exemplu:

ok = True

While ok:

ok = False

for i in range(0, n-1):

If a[i]> a[i+1]:

a[i], a[i+1] = a[i+1], a[i]

ok= True

Caz favorabil: nà ∞; lista este ordonata – T(n) = n

Caz defavorabil: lista este ordonata descrescator- T(n) = n2

Caz mediu: T(n) = n2

Complexitatea generala:

Complexitatea spatiu: O(1)

ok = False

for i in range(0, n):

if a[i]%3 ==0:

ok = True

break

Caz fav: primul elem e divizibil cu 3

T(n) = 1 ϵ Θ(1)

Caz defav: niciun elem nu e divizibil cu 3

T(n) = n ϵ Θ(n)

Caz mediu: T(n) = (1+ 2+3+…n)/n = (n+1) / 2 ϵ Θ(n)

Complexitatea generala: O(n)

Complexitatea spatiu: O(1)

L – lista cu n elemente numere

While len(l)>0 and l[0] % 3 != 0:

l = l[1::] #aici se face o copie sau se foloseste functia copy, functia list (a =[….], b = list(a))

Complexitate timp:

Caz fav: primul elem al listei l se imparte la 3; T(n) = 1 ϵ Θ(1)

Caz defav: niciun elem al listei nu se imparte la 3: T(n) = n2 ϵ Θ(n2)

n2 – pentru ca se face copia element cu element

Caz mediu: T(n) = (1+ 2+3+…n)n/n = n(n+1) / 2 ϵ Θ(n2)

Complex. Generala: O(n2)

Complexitate spatiu:

Caz fav: primul elem al listei l se imparte la 3; T(n) = 1 ϵ Θ(1)

Caz defav: niciun elem al listei nu se imparte la 3: T(n) = n2 ϵ Θ(n2)

Caz mediu: T(n) = (1 \* (n-1) + (n-1) + (n-2) + (n-1) + (n-2) + (n-3) + …. + (n-1) + (n-2) + … (n-n) )/n

= 1 + 2 +3 +…+n ϵ Θ(n2)

Complexitate generala: O(n2)

Lista[n1:n2:n3] à SLICE

N1 – start, de la ce indice iau elem din lista

N2 – end, pana unde sa mearga

N3 – pas

Metode de programare: Backtracking, Divide et impera, programaere dinamica, Greedy

Map, filter, reduce, zip, lambda in Python

add=lambda a,b,c: print(a+b+c)

add(4, 5, 1)

square = list(map(lambda x: x\*\*x, nums))

result = filter(lambda x: x % 2, seq)

result = filter(lambda x: x % 2 == 0, seq)

multiply=reduce(lambda a,b:a\*b,seq)

name = ["Manjeet", "Nikhil", "Shambhavi"]

roll\_no = [4, 1, 3]

marks = [40, 50, 60]

mapped = zip(name, roll\_no, marks)

print(list(mapped))

Output:

[('Manjeet', 4, 40), ('Nikhil', 1, 50), ('Shambhavi', 3, 60)]

name = ["Manjeet", "Nikhil", "Shambhavi"]

marks = [40, 50, 60]

mapped = zip(name, marks)

print(dict(mapped))

Output:

{'Manjeet': 40, 'Nikhil': 50, 'Shambhavi': 60}

Our construction numbers[1:11:2] prints the values between index numbers inclusive of 1 and exclusive of 11, then the stride value of 2 tells the program to print out only every other item.

for x in range(1,4):

sea\_creatures += ['fish']

print(sea\_creatures)

sharks = ['shark']

for x in range(1,4):

sharks \*= 2

print(sharks)

https://www.digitalocean.com/community/tutorials/how-to-use-list-methods-in-python-3

https://www.geeksforgeeks.org/operator-overloading-in-python/

Probleme de eficienta: rezolvare cu dictionar

Generarea permutarilor: se poate rezolva si cu Programare Dinamica :) (https://www.codeproject.com/Tips/891811/Calculating-Permutation-Using-Dynamic-Programming)