Eucscu Irina Ștefania 20.01.2022

Grupa 133

Examen PA

Subiectul I

a)
```
def aparitii (*numere):
    d = {}
    for numar in numere:
        salv = numar
        cifre = []
        d[numar] = []
        while salv != 0:
            cifre.append(salv % 10)
            salv = salv // 10
        cf = set(cifre)
        for cifra in cf:
            tuple = (cifra, cifre.count(cifra))
            d[numar].append(tuple)
    return d
```

b) `lista = [ch for ch in d.keys() if len(d[ch]) >= 3]`

c) $O(n \log_2 n)$

Subiectul II

Folosim metoda backtracking deoarece vrem să rezolvăm o problemă cu multe soluții. Vom verifica soluția progresiv, la fiecare pas al formării ei, evitând astfel generarea tuturor soluțiilor posibile și apoi selectarea celor bune, salvându-se timp.

Variabile: l - lista de litere
s - lista de simboluri
st - stiva în care formăm soluția
niv - nivelul curent al stivei
lit - fiecare literă
simb - fiecare simbol

Enescu Irina Ștefania

Grupa 133

a)
```
def back(niv):
    global l, s, st, n, tipar
    for i in range(n):
        if tipar[i] == 'l':
            for lit in l:
                st[niv] = lit
                if lit not in st[:niv]:
                    if niv == n:
                        print(*st[1:niv+1], sep = '')
                    else:
                        back(niv+1)
        else:
            for simb in s:
                st[niv] = simb
                if simb not in st[:niv]:
                    if niv == n:
                        print(*st[1:niv+1], sep = '')
                    else:
                        back(niv+1)

n = int(input("N: "))
tipar = input("Tipar: ")
l = [litera for litera in input("Litere: ").split()]
s = [simbol for simbol in input("Simboluri: ").split()]
st = [0] * (n+1)
if len(tipar) != n
    print("Imposibil")
else:
    back(1)
```

modificare b)

```
if lit not in st[:niv]
and (niv == 1 and
lit in ['a','A','e','E',
'i','I','o','O','u','U'])
```

Eucscu Irina Stefania

Grupa 133

Subiectul II

```
def ordDesc (elev):
    return int(elev[len(elev)-1])

n = int(input("n="))
h = int(input("h="))
elev = []
for i in range(n):
    elev.append([x for x in (input("Elev: ").split())])
elev.sort(key = ordDesc)
nr = 0
for i in range(len(elev)-1):
    if abs(int(elev[i][len(elev)-1]) - int(elev[i+1][len(elev)-1])) < h:
        print(elev[i][:len(elev)-1], sep = " ", end = ", ")
        print(elev[i+1][:len(elev)-1], sep = " ")
        nr += 1
        i += 1
print(nr)
```

Am folosit tehnica Greedy pentru că problema este o problemă de optimizare, de maximizare.

Complexitatea soluției este $n \log_2 n$, complexitate dată de sortarea datelor după ordinul 8. Cum algoritmul parcurge o singură dată lista cu elevi (cu complexitatea n) și sortarea are complexitate mai mare, complexitatea algoritmului este $n \log_2 n$.

Algoritm: Am creat o listă cu elevi cu numele lor și înălțimea și am sortat-o descrescător după înălțime. Acest lucru ține cont de condiția criteriului de selecție: deoarece înălțimea este descendentă => diferența a

Enescu Maria Ștefania

Grupa 133

doi elevi alăturați este minimă. Parcurgem fiecare elev, iar dacă ~~înălțimile~~ înălțimea lui și a elevului următor respectă condiția formăm o grupă și o numărăm și sărim peste un elev, trecând la următorul elev ce nu e într-o grupă. Dacă nu se respectă condiția, trecem la următorul elev.