Hutan Mihai-Alexandru, grupa 143

Examen la PA

Subiectul I:

a)
```
def palindrom(*cuv):
    d={}
    for c in cuv:
        if c==c[::-1]:
            cs=0
            vc=0
            for i in range(len(c)):
                if c[i] in "aeiou":
                    vc+=1
                else:
                    cs+=1
            ls=[]
            if vc>cs:
                for i in range(len(c)):
                    if c[i] in "aeiou" and c[i] not in ls:
                        ls.append(c[i])
            else:
```

Lițan Mihai-Alexandru, grupa 143

```
    else:
        for i in range(len(c)):
            if c[i] not in "aeiou" and c[i] not in cs:
                cs.append(c[i])
        d[c] = cs
return d
```

b) `numere = [m[i][i] * m[i][i] for i in range(len(m))]`

c) $T(n) = \begin{cases} 1, & n = 3 \\ 3T(n/3) + 1, & n > 3 \end{cases}$ $\quad n = 3^k$

$T_n = 3T\left(\frac{n}{3}\right) + 1 =$

$= 3\left\{3T\left(\frac{n}{9}\right) + 1\right\} + 1 =$

$= 9T\left(\frac{n}{9}\right) + 1 + 3 =$

$\vdots \quad = 3, \text{deoarece } n = 3^k$

$= 3^{k-1} \cdot T\left(\frac{n}{3^{k-1}}\right) + \left(1 + 3 + \ldots + 3^{k-2}\right) =$

$= 3^{k-1} \cdot T(3) + \left(1 + 3 + \ldots + 3^{k-2}\right) =$

$= \frac{1}{3}\log_3 n \cdot 1 + \left(1 + 3 + \ldots + 3^{k-2}\right)$

$\Rightarrow O(\log_3 n)$

2/7

Hutan Mihai-Alexandru, grupa 143

Subiectul 4:

a) Folosesc metoda backtracking, metodă ce îmi generează element cu element în mod recursiv toate soluțiile posibile ce respectă condițiile date.

Reprezentarea soluției se va face în vectorul x, cu elementele $x_0, x_1, \ldots x_{m-1}$.

Fiecare element x_k poate lua valori:

de la 1 la 9 pt. k==0

de la 0 la 9 pt k!=0

Condiții de continuare la pasul k:

- valoarea absolută a diferenței dintre x[k] și x[k-1] să fie în 0 sau 1 (pt. orice k!=0 și k!=m-1)
- pt. k=m-1, la fel ca pt. orice k+ ca valoarea absolută a diferenței dintre $x[\underbrace{k}_{k=m-1}]$ și x[0] să fie 0 sau 1.
- pt. k=0 doar luăm toate cifrele de la 1 la 9

Lutan Mihai-Alexandru, grupa 143

Condiții finale:

~~- să ne aflăm cu k~~

- să ne aflăm pe poziția k == m,

doar atât deoarece condițiile de continuare sunt suficiente

```
def back (k):
 if k == m:
    print(*x, sep='')
 elif k==0:
    for i in range(1, 10):
        x[k] = i
        back (k+1)
 elif k == m-1:
    for i in range (0, 10):                    (pct. 5)
        if abs (i - x[k-1]) <= 1 and abs (i - x[0]) <= 1:
            x[k] = i
            back (k+1)
 else:
    for i in range (0, 10):
        if abs (i - x[k-1]) <= 1:
            x[k] = i
            back (k+1)
```

4/7

Hutan Mihai-Alexandru, grupa 143

```
n = int(input("n = "))
x = [0 for i in range(n)]
back(0)
```

b) unde am notat la subpct. a) cu "(pct. b)"
adaug **and i % 2 == 0**

Subiectul 2:

Folosesc metoda greedy, metodă ce îmi generează soluția pas cu pas, iar la fiecare pas extrag din vectorul sortat elementul care pare cel mai optim în acel moment.

Complexitatea este $n \log_2 n$ deoarece complexitatea funcției sorted este $n \log_2 n$, iar parcurgerile vectorilor sunt de complexitate n, deci ar veni constanta $n \log_2 n$, dar constanta este mică deci rămânem cu $O(n \log_2 n)$

5/5

```
n = int(input("n = "))
g = float(input("g = "))
ob = []
ls = []
sol = []
nl = 0
cn = n
while cn:
    gle = float(input(f"ge. ob. ce nl{n-cn+1}:"))
    ob.append(gle)
    cn -= 1
for i in range(len(ob)):
    ls.appen([i+1])
    ls[i].append(ob[i])
ls = sorted(ls, key= lambda x: x[1])
for i in range(1, len(ls)):
    if abs(ls[i][1] - ls[i-1][1]) <= g:
        sol.append([i])
        sol[nl].append(i-1)
        nl += 1
        ls[i][1] = 999999
```

6/7

```
print (ne)
for x in sd:
    print (*x, sep = "+")
```