Tutoriat 9 – Greedy. Programare dinamică
- rezolvări -

1. Managerul artistic al unui festival de teatru trebuie să selecteze, pentru o singură zi, o mulțime cât mai amplă de spectacole ce pot fi jucate în cele `n` săli pe care le are la dispoziție, sălile fiind numerotate de la `0` la `n-1`. Știind că i s-au propus `m` spectacole de forma `(nr_sala, ora_inceput, minut_inceput, ora_final, minut_final)` și că trebuie să existe o pauză de minim jumătate de oră între spectacole, scrieți un program care să selecteze spectacolele pentru fiecare sală, astfel încât să permită spectatorilor vizionarea unui număr cât mai mare de spectacole.

Exemplu:

Intrare	Iesire
n = 2 m = 7 0 12 30 16 30 0 15 0 18 0 1 17 30 19 0 1 17 30 18 30 0 12 15 13 0 0 13 30 14 0 1 19 15 20 0	0: 12:15-13:00, 13:30-14:00, 15:00-18:00; 1: 17:30-18:30, 19:15-20:00;

```
def strSpectacol( spectacol ):
    return strMinToH( spectacol[0] ) + ' - ' + strMinToH( spectacol[1] )

def strMinToH( ora ):
    minut = str( ora % 60 )
    ora = str( ora // 60 )
    if len( ora ) == 1:
        ora = '0' + ora
    if len( minut ) == 1:
        minut = '0' + minut
    return ora + ':' + minut

n = int( input('n = ') )
m = int( input('m = ') )
sali = [ [] for x in range( n ) ]
for i in range( m ):
    spectacol = [ int(x) for x in input().split() ]
    sali[ spectacol[0] ].append( ( spectacol[1] * 60 + spectacol[2], spectacol[3] *
60 + spectacol[4] ) )

spectacole = [ [] for x in range(n) ]

for i in range( n ):
    sali[i].sort( key = lambda x: x[1] )
```

```
    for spectacol in sali[i]:
        if spectacole[i] == [] or spectacole[i][-1][1] + 30 <= spectacol[0]:
            spectacole[i].append( spectacol )

for i in range( n ):
    print( f'{i}:', ', '.join( [ strSpectacol( spectacol ) for spectacol in
spectacole[i] ] ), end = ';\n' )
```

2. Se citește de la tastatură o listă de numere întregi, reprezentând valorile unor bancnote, și numărul de bucăți disponibile din fiecare bancnotă. Afișați modul în care puteți plăti o sumă **s**, folosind număr minim de bancnote. Tratați și cazul în care suma nu poate fi achitată cu bancnotele disponibile.

Exemplu:

Intrare	Iesire
1 2 3 5 25 8 1 2 2 4 s = 21	5 5 3 3 2 1 1 1
1 1 s = 2	Suma nu poate fi achitata cu bancnotele disponibile.

```
bancnote = [ int(x) for x in input().split() ]
frecvente = [ int(x) for x in input().split() ]
s = int(input('s = '))

v = [ ( bancnote[i], frecvente[i] ) for i in range( len(bancnote) ) ]
v.sort( reverse = True )

sol = []
for pereche in v:
    while pereche[0] <= s and pereche[1] > 0:
        pereche = ( pereche[0], pereche[1] - 1 )
        sol.append( pereche[0] )
        s -= pereche[0]

print( ' '.join( [ str(x) for x in sol ] ) if s == 0 else 'Suma nu poate fi
achitata cu bancnotele disponibile.' )
```

3. Se citesc din fișierul **"matrice.txt"** două numere naturale nenule **n** și **m**, o matrice pătratică de dimensiune **n** și apoi, **m** interogări de forma **(x1,y1,x2,y2)**, reprezentând pozițiile a două elemente din matrice. Afișați, pentru fiecare interogare în parte, suma tuturor elementelor din dreptunghiul determinat de cele două elemente ale matricei.

Exemplu:

matrice.txt	Iesire
5 3 8 3 2 0 4 5 4 9 1 8 2 0 1 5 7 9 9 7 3 2 5 8 6 3 6 0 1 2 3 3 2 3 3 2 0 4 2	25 10 47

```
mat = []
q = []
with open( 'matrice.txt', 'r' ) as f:
    n, m = [ int(x) for x in f.readline().split() ]
    for i in range(n):
        mat.append( [ int(x) for x in f.readline().split() ] )
    for i in range(m):
        q.append( [ int(x) for x in f.readline().split() ] )

s = [ [0] * ( n + 1 ) for i in range( n + 1 ) ]
for i in range( 1, n + 1 ):
    for j in range( 1, n + 1 ):
        s[i][j] = s[i - 1][j] + s[i][j - 1] - s[i - 1][j - 1] + mat[i - 1][j - 1]

for aux in q:
    x1, y1, x2, y2 = aux
    print( s[x2 + 1][y2 + 1] - s[x1][y2 + 1] - s[x2 + 1][y1] + s[x1][y1] )
```

4. În fișierul **"drum.txt"** se află pe fiecare linie **i** câte **i** numere naturale. Ținând cont ca vă puteți deplasa, de pe o linie **i**, pe linia **i+1**, doar în jos sau pe diagonală în dreapta jos, aflați suma maximă a unui drum dintre prima și ultima linie a fișierului și afișați și un drum care are această sumă.

Exemplu:

drum.txt	Iesire
1 3 5 6 4 2 3 1 4 2	14 1 5 4 4

```
def gasesteDrum( i, j ):
    drum.append( mat[i][j] )
    if i == len( mat ) - 1:
        return
    if s[i + 1][j] >= s[i + 1][j + 1]:
        gasesteDrum( i + 1, j )
        return
    gasesteDrum( i + 1, j + 1 )

with open( 'drum.txt', 'r' ) as f:
    mat = [ [ int(x) for x in linie.split() ] for linie in f.readlines() ]

s = [ [0] * i for i in range( 1, len( mat )) ] + [ mat[-1] ]
for i in range( len(s) - 2, -1, -1 ):
    s[i] = [ mat[i][j] + max( s[i + 1][j], s[i + 1][j + 1] ) for j in range( len(
s[i] ) ) ]

drum = []
gasesteDrum( 0, 0 )

print( s[0][0] )
print( *drum )
```

5. Gigel vrea să urce o scară formată din n trepte. Fiind neastâmpărat, el poate sări un număr de trepte mai mic sau egal cu k. În câte moduri poate urca Gigel cele n trepte?

Exemplu:

Intrare	Iesire
n = 3 k = 3	4
Explicatie: Gigel poate urca cele 3 trepte in urmatoarele moduri: 0->1->2->3 0->1->3 0->2->3 0->3	

```
n = int( input('n = ') )
k = int( input('k = ') )

dp = [0] * ( n + 1 )
for i in range( 1, n + 1 ):
    dp[i] = sum( dp[:i] ) + 1 if i <= k else sum( dp[i - k : i] )

print( dp[n] )
```