Programarea Algoritmilor
– SEMINAR NR. 5 –
(grupa 131)

1. **Funcții generice**

Funcție generică ce furnizează numărul elementelor dintr-o listă care au o anumită proprietate (implementată sub forma unei funcții care să returneze True dacă un element are proprietatea cerută sau False în caz contrar).

```
def numarare(lista, proprietate):
    k = 0
    for x in lista:
        if proprietate(x):
            k += 1
    return k
```

Exemple de utilizare a funcției generice:

a) numărul perechilor (x,y) cu proprietatea că x=y

```
def cmpPerechi(t):
    return t[0] == t[1]

n = 4
lp = [(i, j) for i in range(n) for j in range(n)]
print(numarare(lp, cmpPerechi))
```

b) numărul șirurilor de lungime k

```
def cmpLungimeSiruri(k):
    def cmp(s):
        return len(s) == k

    return cmp

lcuv = ["Ana", "are", "mere", "si", "are", "pere"]
print(numarare(lcuv, cmpLungimeSiruri(3)))
```

c) numărul valorilor x dintr-o listă pentru care cmmdc(x,y)=t, unde y și t sunt date

```
def cmpCMMDC(y, t):
    def cmmdc(a, b):
        if b == 0:
            return a
        return cmmdc(b, a % b)

    def cmp(x):
        return cmmdc(x, y) == t

    return cmp

lnr = [10, 172, 23, 17, 18, 100, 13, 15]
print(numarare(lnr, cmpCMMDC(4, 2)))
```

Se va afișa valoarea 2, deoarece două numere din lista lnr au cmmdc dintre ele și 4 egal cu 2 (numerele 10 și 18)

d) sortarea elementelor unei liste după valorile lui cmmdc(elem,y) și apoi după valoarea lui elem

```
def cmpCMMDCSortare(y):
    def cmmdc(a, b):
        if b == 0:
            return a
        return cmmdc(b, a % b)

    def cmp(x):
        return cmmdc(x, y), x

    return cmp

lnr = [10, 172, 23, 17, 18, 100, 13, 15]
lnr.sort(key=cmpCMMDCSortare(30))
print(lnr)
```

2. **Planificarea unor proiecte cu profit maxim**

Se consideră n proiecte, pentru fiecare proiect cunoscându-se profitul, un termen limită (sub forma unei zi din lună) și faptul că implementarea sa durează exact o zi. Să se găsească o modalitate de planificare a unor proiecte astfel încât profitul total să fie maxim.

Exemplu:

proiecte.in			proiecte.out
BlackFace	2	800	Ziua 1: BestJob 900.0
Test2Test	5	700	Ziua 2: FileSeeker 950.0
Java4All	1	150	Ziua 3: JustDoIt 1000.0
BestJob	2	900	Ziua 5: Test2Test 700.0
NiceTry	1	850	
JustDoIt	3	1000	Profit maxim: 3550.0
FileSeeker	3	950	
OzWizard	2	900	

Algoritm de complexitate $O(n^2)$:

```
def cmp_profit_proiect(p):
    return p[2]

def cmp_data_proiect(p):
    return p[1]

fin = open("proiecte.in")
lsp = []
for linie in fin:
    aux = linie.split()
    lsp.append((aux[0], int(aux[1]), float(aux[2])))
fin.close()

lsp.sort(key=cmp_profit_proiect, reverse=True)

data_max = max(lsp, key=cmp_data_proiect)[1]

programare = {k: None for k in range(1, data_max + 1)}

profitmax = 0
for proiect in lsp:
    for z in range(proiect[1], 0, -1):
        if programare[z] == None:
            programare[z] = proiect
            profitmax += proiect[2]
            break
```

```
fout = open("proiecte.out", "w")

for z in programare:
    if programare[z] != None:
        fout.write("Ziua " + str(z) + ": " + programare[z][0] + " " +
                   str(programare[z][2]) + "\n")
fout.write("\nProfit maxim: " + str(profitmax))

fout.close()
```

Observație: Pentru algoritmul de complexitate O(n*log(n)) vezi seminarul 6.

TEMĂ: Programarea proiectelor să se facă fără zile libere între proiecte.

3. Se citesc n intervale închise. Să se calculeze suma lungimilor lor.
Exemplu:

intervale.in	intervale.out
570 670	435
500 590	
600 680	
690 840	
930 1005	
730 790	
700 795	
900 960	

Algoritm de complexitate O(n*$\log_2$n):

```
def cmp_intervale(t):
    return t[0], -t[1]

f = open("intervale.in")

intervale = []
for linie in f:
    aux = linie.split()
    intervale.append((int(aux[0]), int(aux[1])))

f.close()

intervale.sort(key=cmp_intervale)

minr = intervale[0][0]
maxr = intervale[0][1]
total = maxr - minr

for i in range(1, len(intervale)):
    if intervale[i][1] <= maxr:
        continue
    elif intervale[i][0] < maxr:
        total += intervale[i][1] - maxr
        maxr = intervale[i][1]
    else:
        total += intervale[i][1] - intervale[i][0]
        minr = intervale[i][0]
        maxr = intervale[i][1]

fout = open("intervale.out", "w")
fout.write(str(total))
fout.close()
```

TEMĂ: Modificați programul de mai sus astfel încât să afișeze și reuniunea intervalelor date. Pentru exemplul din enunț, reuniunea lor este [500,680]U[690,840]U[900,1005].

4. **Problema mulțimii minime de acoperire / Problema cuielor**
   Fie n intervale $[a_i, b_i]$. Să se determine o mulțime M cu număr minim de elemente astfel încât $\forall i \in \{1,2,\dots,n\}, \exists x \in M$ astfel încât $x \in [a_i, b_i]$.

   Exemplu: Pentru intervalele din exemplul dat la problema 3, o mulțime de acoperire a lor este M={590, 680, 790, 960}.

   Algoritm de complexitate $O(n*\log_2 n)$:

```
def cmpIntervale(interval):
    return interval[1]

fin = open("intervale.in")
intervale = []
for linie in fin:
    aux = linie.split()
    intervale.append((int(aux[0]), int(aux[1])))
fin.close()

intervale.sort(key=cmpIntervale)

ult = 0
for icrt in intervale:
    if icrt[0] > ult:
        print(icrt[1])
        ult = icrt[1]
```