Programarea Algoritmilor **Universitatea din București**
Grupa 151 **Facultatea de Matematică și Informatică**

Laborator 5

Tehnici de programare. Greedy

1. Problema rucsacului

Se consideră n obiecte, pentru fiecare cunoscând valoarea și greutatea sa. Având la dispoziție un rucsac în care se pot încărca obiecte (sau fragmente de obiecte) în limita unei greutăți maxime date, să se determine o încărcare optimală a rucsacului, respectiv valoarea totală a obiectelor din rucsac să fie maximă.

Fișierul *obiecte.txt* are pe prima linie numărul de obiecte n, pe următoarele n linii câte două numere (valoarea și greutatea obiectului curent), iar la final greutatea maximă care se poate încărca în rucsac. În fișierul *rucsac.txt* să se afișeze obiectele încărcate în rucsac, precum și valoarea acestora.

Indicație de rezolvare:

1. Se sortează obiectele descrescător după raportul dintre valoare și greutate.
2. Parcurgând lista sortată, cât timp greutatea disponibilă rămasă în rucsac este mai mare sau egală decât greutatea obiectului curent, se adaugă acest obiect întreg în rucsac. Apoi, dacă se poate, se adaugă un anumit procent p din obiectul următor, până se atinge greutatea totală maximă. Valoarea totală se calculează adunând valorile obiectelor întregi din rucsac, plus procentul p din valoarea obiectului parțial, dacă există.

Exemplu:

obiecte.txt	**rucsac.txt**
3	Greutate:
120 30	50 = 10 + 20 + (2/3) * 30
60 10	
100 20	Valoarea obiectelor din rucsac:
50	60 + 100 + (2/3) * 120 = 240

2. Minimizarea timpului mediu de așteptare

Din fișierul *tis.txt* se citesc numere naturale nenule reprezentând timpii necesari pentru servirea fiecărei persoane care așteaptă la o coadă. Să se determine ordinea în care ar trebui servite persoanele de la coadă astfel încât timpul mediu de așteptare să fie minim.

Indicație de rezolvare:

1. Se creează o listă de tupluri care să conțină, pentru fiecare persoană, numărul său de ordine în lista inițială și timpul individual de servire pentru acea persoană.
2. Se definește o funcție *afisare_timpi_servire* care primește ca parametru o listă de tupluri de tipul indicat mai sus și afișează pentru fiecare persoană, pe 3 coloane, următoarele informații: numărul de ordine al persoanei, timpul individual de servire și timpul său de așteptare. La final se afișează timpul mediu de așteptare al tuturor persoanelor.
3. Se apelează funcția de mai sus pentru lista inițială, iar apoi pentru lista care a fost sortată crescător după timpul individual de servire.

Exemplu:
Dacă *tis.txt* conține numerele **7 15 3 7 8 3 2 10 5 4 2**, atunci timpul mediu de așteptare inițial va fi: **41.73**, iar timpul mediu de așteptare după sortarea listei va fi: **24.82**

3. Programarea spectacolelor

Fișierul *spectacole.txt* conține, pe câte un rând, ora de început, ora de sfârșit și numele câte unui spectacol. Să se creeze o listă care să conțină, în tupluri formate din câte 3 șiruri de caractere, cele 3 informații despre fiecare spectacol. Să se programeze într-o singură sală un număr maxim de spectacole care să nu se suprapună. În fișierul *programare.txt* să se afișeze, pe câte un rând, intervalul de desfășurare și numele pentru spectacolele selectate.

Indicație de rezolvare:

1. Se sortează lista de spectacole crescător după ora de sfârșit.
2. Parcurgând lista sortată, se alege câte un spectacol astfel încât să nu se suprapună cu ultimul spectacol ales (ora de început să fie mai mare sau egală decât ora de sfârșit a ultimului ales).

Exemplu:

spectacole.txt

10:00-11:20 Scufita Rosie
09:30-12:10 Punguta cu doi bani
08:20-09:50 Vrajitorul din Oz
11:30-14:00 Capra cu trei iezi
12:10-13:10 Micul Print
14:00-16:00 Povestea porcului
15:00-15:30 Frumoasa din padurea adormita

programare.txt

08:20-09:50 Vrajitorul din Oz
10:00-11:20 Scufita Rosie
12:10-13:10 Micul Print
15:00-15:30 Frumoasa din padurea adormita

4. Turn de înălțime maximă format din cuburi

Se dă o mulțime de n cuburi. Fiecare cub este caracterizat prin lungimea laturii și culoare. Nu există două cuburi având aceeași dimensiune. Fișierul *cuburi.txt* conține pe prima linie un număr natural nenul n (numărul de cuburi), apoi pe următoarele n linii câte un număr natural nenul (lungimea laturii cubului) și un șir de caractere (culoarea cubului).

Să se construiască un turn de înălțime maximă astfel încât peste un cub cu latura L și culoarea K se poate așeza doar un cub cu latura mai mică strict decât L și culoare diferită de K. În fișierul *turn.txt* să se afișeze componența turnului de la bază spre vârf, pe câte un rând latura și culoarea cubului, apoi la final să se afișeze înălțimea totală a turnului.

Indicație de rezolvare:

1. Se sortează lista de cuburi descrescător după lungimea laturii.
2. La baza turnului se așază cubul de latură maximă, iar apoi se parcurge lista sortată și se adaugă la turn câte un cub care are culoare diferită de ultimul cub adăugat.

Exemplu:

cuburi.txt

6
7 verde
10 rosu
6 albastru
12 rosu
4 rosu
9 verde

turn.txt

12 rosu
9 verde
6 albastru
4 rosu

Inaltime totala: 31

5. Plata unei sume folosind un număr minim de bancnote

Fie o mulțime de bancnote $\{B_0, B_1, \dots B_n\}$ astfel încât $B_0 = 1$ (avem mereu bancnota unitate, pentru a putea plăti orice sumă) și $B_i \mid B_j, \forall 0 \leq i < j \leq n-2$ (cu excepția ultimelor 2 bancnote, toate valorile se divid cu toate valorile din listă mai mici decât ele). De exemplu se consideră bancnotele românești cu valorile $1, 5, 10, 50, 100, 200, 500$.

Pentru o sumă de bani S, să se determine o modalitate de a plăti suma S folosind un număr minim de bancnote. Fișierul *bani.txt* conține pe prima linie valorile bancnotelor disponibile (se consideră că avem la dispoziție un număr infinit din fiecare bancnotă), iar pe a doua linie valoarea sumei S. În fișierul *plata.txt* să se afișeze ce bancnote cu valori diferite și câte din fiecare valoare s-au folosit pentru a achita suma S.

Indicație de rezolvare:

1. Se sortează lista de bancnote în ordine descrescătoare.
2. Parcurgând lista cât timp suma rămasă de plată nu este nulă, se alege acea bancnotă cu valoarea mai mică sau egală decât suma rămasă de plată și se scade (de numărul maxim posibil e ori) valoarea bancnotei din suma rămasă.

Exemplu:

bani.txt

```
1 5 10 50 100 200 500
173
```

plata.txt

```
173 = 100 * 1 + 50 * 1 + 10 * 2 + 1 * 3
```

6. Minimizarea întârzierii maxime a unor activități

Fie o mulțime de n activități. Fiecare activitate are o anumită durată d_i (se execută într-un număr de unități de timp) și un termen limită t_i până la care ar trebui executată (un număr de unități de timp indicat de la începutul programării tuturor activităților $t_0 = 0$). O activitate care începe la momentul s_i și se termină la momentul $f_i = s_i + d_i$ are o întârziere de $h_i = \max\{0, f_i - t_i\}$ unități de timp. Se cere programarea activităților astfel încât să se minimizeze întârzierea maximă $H = \max(h_i)$.

Fișierul *activitati.txt* conține pe prima linie numărul de activități n, iar pe următoarele n linii conține câte două numere indicând durata și termenul limită al fiecărei activități. Să se afișeze în fișierul *intarzieri.txt* programarea activităților, pe câte o linie, astfel: intervalul ales (s_i–>f_i), de lungime egală cu durata d_i, termenul limită t_i și întârzierea h_i pentru fiecare activitate. Pe ultima linie din fișier să se afișeze întârzierea maximă.

Indicație de rezolvare:

1. Se sortează lista crescător după termenul limită t.
2. Se inițializează timpul curent cu $T = 0$. Se parcurge lista și se programează activitatea curentă în intervalul (T–>$T + d$) și se actualizează $T = T + d$, după se trece la următoarea activitate din listă.

Exemplu:

activitati.txt

```
6
1 9
3 14
2 8
2 15
3 6
4 9
```

intarzieri.txt

Interval	Termen	Intarziere
(0 –> 3)	6	0
(3 –> 5)	8	0
(5 –> 6)	9	0
(6 –> 10)	9	1
(10 –> 13)	14	0
(13 –> 15)	15	0

Intarzierea maxima: 1

7. Planificarea proiectelor cu profit maxim

Se consideră o mulțime de proiecte, fiecare având un termen limită și un profit asociat dacă proiectul este terminat până la termenul limită. Fiecare proiect este realizat într-o singură unitate de timp. Să se planifice proiectele (fără a se suprapune ca timp) astfel încât să se maximizeze profitul total.

Fișierul *proiecte.txt* conține, pe fiecare linie, numele, termenul limită și profitul asociat unui proiect. În fișierul *profit.txt* să se afișeze succesiunea de proiecte alese și profitul total obținut prin realizarea lor.

Indicație de rezolvare:

1. Se sortează lista de proiecte descrescător după profit.
2. Folosind un dicționar care conține un număr de intrări egal cu maximul termenelor limită, se va încerca planificarea fiecărui proiect cât mai aproape de termenul său limită.

Exemplu:

proiecte.txt	**profit.txt**
a 2 100 b 1 19 c 2 27 d 1 25 e 3 15	T = 3 Proiecte: c, a, e Profit: 27+100+15 = 142

8. Numărul minim de săli necesare pentru a programa mai multe spectacole

Fișierul *spectacole.txt* conține, pe câte un rând, ora de început, ora de sfârșit și numele câte unui spectacol. Să se creeze o listă care să conțină, în tupluri formate din câte 3 șiruri de caractere, cele 3 informații despre fiecare spectacol. Să se determine numărul minim de săli necesare k pentru a putea programa toate spectacolele, fără să existe suprapuneri între spectacolele din aceeași sală. În fișierul *sali.txt* să se afișeze k și apoi spectacolele care au fost programate în fiecare dintre cele k săli.

Indicație de rezolvare:

1. Se sortează lista de spectacole crescător după ora de început.
2. Parcurgând lista sortată, se alege câte un spectacol și se programează în oricare dintre sălile disponibile (dacă spectacolul curent începe după ora de sfârșit a ultimului spectacol din acea sală) sau se programează într-o nouă sală (dacă în toate sălile disponibile există deja spectacole care se suprapun cu spectacolul curent).

Exemplu:

proiecte.txt	**profit.txt**
15:00-16:30 j 11:00-12:30 d 09:00-10:30 a 13:00-14:30 f 14:00-16:30 h 11:00-14:00 e 15:00-16:30 i 09:00-12:30 b 13:00-14:30 g 09:00-10:30 c	3 sali (09:00-10:30 a), (11:00-12:30 d), (13:00-14:30 f), (15:00-16:30 j) (09:00-12:30 b), (13:00-14:30 g), (15:00-16:30 i) (09:00-10:30 c), (11:00-14:00 e), (14:00-16:30 h)