Programarea Algoritmilor
– LABORATOR NR. 5 –

Tehnica de programare Greedy

1. Minimizarea timpului mediu de așteptare ***[Curs 9, pag 3]***

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:

- 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.
- Se definește o funcție "afisare_timpi_servire (tis)" 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.
- 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**

2. Planificarea optimă a unor spectacole într-o singură sală ***[Curs 10, pag 1]***

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:

- Se sortează lista de spectacole *crescător după ora de sfârșit*.
- 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

3. 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:

- Se sortează lista de cuburi *descrescător după lungimea laturii*.
- 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	**turn.txt**
6	12 rosu
7 verde	9 verde
10 rosu	6 albastru
6 albastru	4 rosu
12 rosu	
4 rosu	Inaltime totala: 31
9 verde	

4. Plata unei sume folosind un număr minim de bancnote ***[Curs 9, pag 2]***

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 $B_i | B_j, \forall\ 0 \leq i < j \leq n-2$ (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:

- Se sortează lista de bancnote în ordine *descrescătoare*.
- 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 de ori) valoarea bancnotei din suma rămasă.

Exemplu:

bani.txt	**plata.txt**
1 5 10 50 100 200 500	173 = 100*1 + 50*1 + 10*2 + 1*3
173	

5. Minimizarea întârzierii maxime a unor activități

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

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:

- Se sortează lista *crescător după termenul limită* ***t***.
- 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

6. Planificarea unor spectacole folosind un număr minim de săli ***[Curs 10, pag 5]***

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:

- Se sortează lista de spectacole *crescător după ora de început*.
- 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:

evenimente.txt	**sali.txt**
15:00-16:30 j	Numar minim de sali: 3
11:00-12:30 d	
09:00-10:30 a	Sala 1:
13:00-14:30 f	(09:00-10:30 a), (11:00-12:30 d), (13:00-14:30 f), (15:00-16:30 j)
14:00-16:30 h	
11:00-14:00 e	Sala 2:
15:00-16:30 i	(09:00-12:30 b), (13:00-14:30 g), (15:00-16:30 i)
09:00-12:30 b	
13:00-14:30 g	Sala 3:
09:00-10:30 c	(09:00-10:30 c), (11:00-14:00 e), (14:00-16:30 h)

7. Problema rucsacului (varianta continuă/fracționară) ***[Curs 10, pag 11]***

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 greutatea maximă care se poate încărca în rucsac, iar pe următoarele linii câte două numere reale (valoarea și greutatea obiectului curent). În fișierul "rucsac.txt" să se afișeze câștigul maxim și obiectele încărcate în rucsac.

Indicație de rezolvare:

- Se sortează obiectele *descrescător după raportul dintre valoare și greutate*.
- 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**
53	Castig maxim: 134.0
10 30	
5 40	Obiectele incarcate:
18 36	Obiect 2: 100.00%
20 10	Obiect 1: 100.00%
8 16	Obiect 3: 100.00%
40 30	Obiect 5: 100.00%
20 20	Obiect 7: 60.00%

8. Planificarea unor proiecte cu profit maxim ***[Seminar 5, pag 11]***

Se consideră o mulțime de proiecte, fiecare având un termen limită (sub forma unei zile din lună) și un profit asociat dacă proiectul este terminat până la termenul limită. Fiecare proiect durează exact o zi. 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:

- Se sortează lista de proiecte *descrescător după profit*.
- 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	Ziua 1: c
b 1 19	Ziua 2: a
c 2 27	Ziua 4: e
d 1 25	
e 4 15	Profit maxim: 27+100+15 = 142

9. Problema mulțimii de acoperire / Problema cuielor ***[Seminar 5, pag 6]***

Fie n intervale închise $I_1 = [a_1, b_1], \dots, I_n = [a_n, b_n]$. Să se determine o mulțime M cu număr minim de elemente astfel încât $\forall k \in \overline{1,n}, \exists x \in M$ astfel încât $x \in I_k = [a_k, b_k]$. Mulțimea M se numește *mulțime de acoperire* a șirului de intervale respectiv.

(***Problema cuielor*****:** Se consideră n scânduri, fiecare fiind dată printr-un interval închis de pe axa reală. Să se determine numărul minim de cuie pe care trebuie să le batem astfel încât în fiecare scândură să fie bătut cel puțin un cui. Se consideră faptul că orice cui are o lungime suficient de mare pentru a trece prin oricâte scânduri este nevoie.)

Fișierul "intervale.txt" conține, pe fiecare linie, câte două numere naturale reprezentând capetele unui interval (capetele unei scânduri). În fișierul "acoperire.txt" să se afișeze elementele mulțimii de acoperire (locațiile în care se bat cuiele).

Indicație de rezolvare:

- Se sortează lista de intervale *crescător după capătul din dreapta*.
- Se bate primul cui la capătul drept al primei scânduri. Pentru fiecare dintre celelalte scânduri, dacă nu conține niciunul dintre cuiele bătute anterior, atunci se bate un nou cui la capătul ei drept.

Exemplu:

intervale.txt	**acoperire.txt**
570 670	590
500 590	680
600 680	790
690 840	930
730 790	
700 800	
900 930	

10. Reuniunea intervalelor ***[Seminar 5, pag 7]*** *(Nu e algoritm de tip Greedy !)*

Fie n intervale închise $I_1 = [a_1, b_1], \dots, I_n = [a_n, b_n]$. Să se determine reuniunea intervalelor date, precum si lungimea sa.

Fișierul "intervale.txt" conține, pe fiecare linie, câte două numere naturale reprezentând capetele unui interval. În fișierul "reuniune.txt" să se afișeze reuniunea intervalelor și lungimea sa.

Indicație de rezolvare:

- Se sortează lista de intervale *crescător după capătul din stânga* și, în caz de egalitate, *descrescător după capătul din dreapta*.

Exemplu:

intervale.txt	**reuniune.txt**
570 670	Reuniunea intervalelor:
500 590	[500, 680]
600 680	[690, 840]
690 840	[900, 930]
730 790	
700 800	Lungimea reuniunii: 360
900 930	

11. Interclasarea optimă a *n* vectori ordonați

Se dau lungimile a *n* vectori ordonați crescător L1, L2, …, Ln. Se dorește obținerea unui vector ordonat crescător care conține toate elementele celor *n* vectori inițiali, interclasând succesiv perechi de vectori. Știind că interclasarea a doi vectori de lungimi A respectiv B necesită A+B deplasări, să se determine o ordine în care trebuie să se realizeze interclasările astfel încât numărul total de deplasări să fie minim.

Fișierul "lungimi.txt" conține, pe fiecare linie, numele vectorului și un număr natural reprezentând lungimea sa. În fișierul "interclasari.txt" să se afișeze ordinea în care se fac interclasările, precizând la fiecare pas numele vectorului obținut după interclasare (numele este de forma L_{n+i} unde i este numărul de ordine al interclasării curente) și numărului de deplasări făcute. La final să se afișeze numărul total de deplasări făcute.

Indicație de rezolvare:

- Se reține într-o structură vectorii (având lungimea și numele lor) astfel încât să fie sortați *crescător după lungime*.
- La fiecare pas (cât timp există cel puțin doi vectori), se extrag din structură cei mai scurți doi vectori, se calculează lungimea și numele noului vector (suma lungimilor si L{nr_vector}), care apoi se inserează în structură astfel încât aceasta să rămână sortată corespunzător.
- Pentru a obține complexitatea optimă $\mathcal{O}(n \cdot log(n))$, se salvează tuplurile (lungime, nume) într-o **coadă cu priorități (priority queue)** ***[vezi Curs 10, pag 9]***.

Exemplu:

lungimi.txt	**interclasari.txt**
L1 20 L2 30 L3 20 L4 35 L5 45	Pas 1: Structura contine: (20, L1), (20, L3), (30, L2), (35, L4), (45, L5). Din vectorii (20, L1) si (20, L3) rezulta (40, L6). Pas 2: Structura contine: (30, L2), (35, L4), (40, L6), (45, L5). Din vectorii (30, L2) si (35, L4) rezulta (65, L7). Pas 3: Structura contine: (40, L6), (45, L5), (65, L7). Din vectorii (40, L6) si (45, L5) rezulta (85, L8). Pas 4: Structura contine: (65, L7), (85, L8). Din vectorii (65, L7) si (85, L8) rezulta (150, L9). Pas5: Structura contine: (150, L9). Numar total deplasari: 40 + 65 + 85 + 150 = 340.