Programarea algoritmilor
– SEMINAR NR. 7 –

1. Descompunerea unui număr natural ca sumă de numere naturale nenule.

Variante (exemple pentru n = 4)**:**

a) toate descompunerile posibile (1+1+1+1, 1+1+2, 1+2+1, 1+3, 2+1+1, 2+2, 3+1, 4)
b) descompuneri distincte, respectiv nu au aceiași termeni în altă ordine (1+1+1+1, 1+1+2, 1+3, 2+2, 4) => elementele soluției vor fi păstrate în ordine crescătoare => vom inițializa elementul curent sol[k] al soluției cu valoarea elementului anterior sol[k-1]
c) descompuneri distincte cu termeni distincți (1+3, 4) => elementele soluției vor fi păstrate în ordine strict crescătoare:
d) descompuneri cu termeni distincți (1+3, 3+1, 4) => verificăm $X[k] \notin X[0:k]$ (la fel ca în problema generării permutărilor):
e) soluție cu lungime dată: $\leq p, == p, \geq p$ (pentru $p = 3$: 1+1+2, 1+2+1, 2+1+1)

2. Să se afișeze toate numerele naturale având cifre distincte și suma cifrelor egală cu o valoare c dată. De exemplu, pentru $c = 3$, trebuie să fie afișate numerele: 102, 12, 120, 201, 21, 210, 3 și 30 (nu neapărat în această ordine).

3. Se consideră n spectacole pentru care se cunosc intervalele de desfășurare. Să se găsească toate planificările cu număr maxim de spectacole care se pot efectua într-o singură sală astfel încât, în cadrul fiecărei planificări, spectacolele să nu se suprapună.

Exemplu:

spectacole.in	**spectacole.out**
10:00-11:20 Scufita Rosie	08:20-09:50 Vrajitorul din Oz
09:30-12:10 Punguta cu doi bani	10:00-11:20 Scufita Rosie
08:20-09:50 Vrajitorul din Oz	12:10-13:10 Micul Print
11:30-14:00 Capra cu trei iezi	15:00-15:30 Frumoasa si Bestia
12:10-13:10 Micul Print	
14:00-16:00 Povestea porcului	08:20-09:50 Vrajitorul din Oz
15:00-15:30 Frumoasa si Bestia	10:00-11:20 Scufita Rosie
	12:10-13:10 Micul Print
	14:00-16:00 Povestea porcului
	08:20-09:50 Vrajitorul din Oz
	10:00-11:20 Scufita Rosie
	11:30-14:00 Capra cu trei iezi
	15:00-15:30 Frumoasa si Bestia
	08:20-09:50 Vrajitorul din Oz
	10:00-11:20 Scufita Rosie
	11:30-14:00 Capra cu trei iezi
	14:00-16:00 Povestea porcului

4. Se dau n perechi (x, y) cu proprietatea că x < y. Se cere lungimea maximă k a unui lanț de perechi de forma (x_1, y_1),…, (x_i, y_i), (x_{i+1}, y_{i+1}), …, (x_k, y_k) astfel încât $y_i < x_{i+1}$.

Exemplu: Pentru n = 6 și lp = [(12, 15), (5, 7), (20, 30), (6, 8), (9, 11), (13, 18)], lungimea maximă k a unui lanț cu proprietatea cerută este egală cu 4, iar un posibil astfel de lanț este (5, 7), (9, 11), (12, 15), (20, 30). Atenție, soluția nu este unică!

5. Partiționarea unei mulțimi în două submulțimi cu sume cât mai apropiate (problemă NP-completă)

Exemplu: A = [5, 3, 5, 4, 5] => A1 = [5, 5], A2 = [5, 4, 3] (suma(A1) = 10, suma(A2) = 12 => diferența minimă = 2)

6. Planificarea proiectelor cu bonus maxim

Considerăm n proiecte $P_1, P_2, \dots, P_n$ pe care poate să le execute o echipă de programatori într-o anumită perioadă de timp (de exemplu, o lună), iar pentru fiecare proiect se cunoaște un interval de timp în care acesta trebuie executat (exprimat prin numerele de ordine a două zile din perioada respectivă), precum și bonusul pe care îl va obține echipa dacă proiectul este finalizat la timp (altfel, echipa nu va obține niciun bonus pentru proiectul respectiv). Să se determine o modalitate de planificare a unor proiecte care nu se suprapun astfel încât bonusul obținut de echipă să fie maxim. Vom considera faptul că un proiect care începe într-o anumită zi nu se suprapune cu un proiect care se termină în aceeași zi!

Exemplu:

proiecte.in	**proiecte.out**
P1 7 13 850	P4: 02-06 -> 650 RON
P2 4 12 800	P1: 07-13 -> 850 RON
P3 1 3 250	P5: 13-18 -> 1000 RON
P4 2 6 650	P7: 25-27 -> 300 RON
P5 13 18 1000	
P6 4 16 900	Bonusul echipei: 2800 RON
P7 25 27 300	
P8 15 22 900	