Suplimentar – se pot prezenta maxim 3 probleme (dacă punctajul la celelalte teme≥8.5)

1. Se dă un șir de n numere. Să se determine un cel mai lung subșir crescător (strict) al acestuia **O(n log n)** (**1p**)
2. Se consideră o secvenţă de cărţi de joc luate **din mai multe pachete de cărţi**, aşezate într-un teanc pe masă (carţile sunt date în secvenţă de la bază la vârf). O carte dintr-un pachet de joc va fi reprezentată ca o pereche (număr, litera), unde numele sunt de la 2 la 14 (11 pentru as, 12 pentru juvete, etc..) iar litera corespunde unui simbol (D – diamonds, H – hearts, C – clubs, S – spades). Jokerii sunt reprezentați doar prin litera „J”. Din această secvenţă de cărţi dorim să extragem un număr maxim de cărţi, respectând următoarele reguli: fiecare carte extrasă este situată deasupra cărţii extrase anterior (astfel, cărţile extrase formează un subşir al secvenţei date), iar cartea nou extrasă şi cea extrasă la pasul anterior trebuie să aibă același număr sau să aibă același simbol sau una dintre ele să fie un Joker.
   a) Să se afişeze un astfel de subşir de lungime maximă care poate fi extras din secvenţa dată. (**O(n)**)
   Exemplu – pentru secvența următoare am subliniat elementele dintr-un astfel de sir de lungime maximă:

   8 D 7 S 5 S 8 H 6 H J 5 D 11 S 11 D 12 D

   b) Se folosește un pachet ”special” de cărți, ce conține cărți numerotate de la 1 la n (nu doar pana la 14), cu n citit la intrare. Secvența dată la intrare este formată folosind elemente dintr-un singur astfel de pachet de cărți. Să se găsească un șir de lungime maximă cu proprietatea anterior descrisă (**$O(n^2)$**) (**2,5p**)
3. **Maximum Weighted Independent Set in a Tree (v. curs)** Se consideră un arbore cu mulţimea vârfurilor V = {v_1,..., v_n}. Vârfurile grafului au asociate ponderile w_1,..., respectiv w_n. Să se determine o mulțime independentă de vârfuri de pondere maximă (făcut la curs pentru arbore de tip lanţ) – O(n). Se poate generaliza ideea pentru a rezolva polinomial aceeaşi problemă pentru grafuri neorientate oarecare? **(2p)**
4. Dat n, să se determine numărul de şiruri de lungime n peste alfabetul {1,2,3} care respectă constrângerile:
   - orice 1 are pe poziţiile alăturate în stânga şi în dreapta valoarea 3
   - orice 2 are pe cel puțin una dintre poziţiile din stânga valoarea 3 (**v. curs**) **O(n) (1p)**
5. Se dau două șiruri s şi t de lungimi n, respectiv m. Să se numere de câte ori apare t ca subșir al lui s. Exemplu: s = “abxcaybc” t= “abc” - sunt 4 subșiruri ale lui s egale cu t: {s[1],s[2],s[4]}, {s[1],s[2], [8]}, {s[1],s[7],s[8]}, {s[5],s[7],s[8]}. **O(nm) (1,5p)**

6. Se consideră alfabetul A format din literele mici începând cu ‘a’ şi terminând cu litera mică *c* citită de la intrare. Pe acest alfabet este dată o lege de compoziţie printr-o matrice t (nu se presupune că legea este asociativă sau comutativă). Fiind dat un cuvânt x de lungime n peste alfabetul A şi o literă *dest* din A, să se determine dacă există o parantezare a literelor cuvântului x astfel încât rezultatul efectuării calculelor să fie *dest*. **$O(n^3)$ (3p)**

7. https://www.topcoder.com/community/data-science/data-science-tutorials/dynamic-programming-from-novice-to-advanced/ (problema **upper intermediate**) Se dau un graf neorientat ponderat, câte un cost c_i asociat fiecărui vârf i şi o sumă M de bani pe care o avem la dispoziţie. Pentru a trece prin vârful i trebuie plătită suma c_i. Dacă nu avem suficienţi bani nu putem trece prin acest vârf. Să se determine un drum minim de la 1 la n în aceste condiţii (pe care îl putem parcurge având la dispoziţie suma M). Dacă există mai multe astfel de drumuri, să se afişeze cel mai ieftin. **(3,5p)**