Tema 4 - Heap-uri. Cozi de prioritate

1. **Sortare** Să se implementeze algoritmul **Heap-Sort**. Să se sorteze un vector de numere. (1p)

2. **Prority queue**. Implementaţi o coadă de priorităţi folosind o structură (clasă) PRIORITY_QUEUE, care să aibă un câmp DATA de tip vector de întregi, care să stocheze elementele cozii sub forma unui heap max şi un câmp SIZE - nr. de elemente stocate în coadă. În plus structura trebuie să aibă metodele:

   - INSERT - inserează un nou nod în coadă (0.5 p)
   - EXTRACT_MAX - extrage elementul de prioritate maximă din coadă (0.5 p)
   - MAX_ELEMENT - returnează elementul de prioritate maximă (0.25 p)
   - INCREASE_KEY - creşte prioritatea unui nod.(0.5 p)
   - MAX_HEAPFY (sau SIFT_DOW) - funcţia care coboară o cheie pe poziţia corespunzătoare din heap. (0.5 p)

   În funcţia *main* se declară o variabilă de tip PRIORITY_QUEUE şi se foloseşte un *menu* implementat cu ajutorul unei instrucţiuni *switch*, prin care utilizatorul să poată selecta oricare dintre operaţiile de inserţie, extragerea maximului, obţinerea maximului şi afişarea elementelor din heap. (0.75 p)

3. **Problema câstigului maxim**. Se consideră o sală de spectacole cu M rânduri. Fiecare rând i are $Nr[i]$ locuri. Preţul unui loc depinde de numărul de locuri libere pe rândul respectiv şi de poziţia rândului şi este egal cu $liberere(i) * i$. Evident, iniţial nunmărul de locuri libere de pe un rând i este egal cu $Nr[i]$. Vânzătorul de bilete vinde biletele, aşa încât so obţină un profit maxim de pe urma spectatorilor (se presupune că numărul de spectatori este mereu mai mic decât numărul de locuri). Emil a cumpărat al N-lea bilet. Cât va plăti pe el? (3p)

4. **Problema medianului**. Se consideră un vector de n numere întregi. Se numeşte **median** acel element din vector, care s-ar afla pe poziţia din mijloc în

vectorul sortat. Dacă n este număr par, atunci medianul este media artimetică a celor 2 numere aflate pe poziţiile din centru.
Exemplu: pentru $v = \{6, 1, 5, 2, 0\}$ medianul este 2. Pentru $v = \{4, 1, 5, 2, 6, 3\}$ medianul este 3.5.
Să se citească pe rând dintr-un fişier n numere şi să se afişeze după fiecare citire medianul curent al numerelor citite. **Exemplu**: se citesc din fişier elementele 5, 2, 8, 6, 3 ,2, 9. Se vor afişa: 5, 3.5, 5, 5.5, 5, 4, 5.
NU se sortează numerele. (4p)

5. **Codificarea Huffmann**. Se citeşte un text dintr-un fişier. Să se construiască arborele de codificare Huffmann corespunzător. Să se afişeze codul corespunzător fiecărui caracter şi să se codifica textul. Utilizaţi o coadă de prioritate (min). (5p)

6. **Interclasaea optimală**. Se consideră n vectori de numere întregi sortaţi crescător. Să se interclasesez aceşti vectori într-unul singur cu număr minim de comparaţii. **Observaţie**: se obţine număr minim de comparaţii, dacă la fiecare moment dat se interclasează cei mai scurţi doi vectori. (3p)

7. Să se rezolve problema labirintului de la tema 2 utilizând algoritmul A* (5p). Utilizaţi ca şi funcţie de cost **city-block distance**, adică $|x_1 - x_2| + |y_1 - y_2|$.

Observaţii:

1. Rezolvările problemelor începând de la 3 presupun utilizarea de cozi de prioritate. Nu se vor cumula punctele cu punctele de la pb. 2.

2. Pentru problemele începând de la 3 trebuie implementată complet coada de priorităţi, pentru cel puţin una dintre probleme. Pentru celelalte poate fi utilizată cea implementată în STL. (în cazul în care va apucaţi de aceste probleme).

3. Pentru A* vă recomand următorul site:
https://www.redblobgames.com/pathfinding/a-star/introduction.html

4. Cine predă doar primele 2 pb trebuie sa ştie cum funcţionează o coadă de priorităţi şi va fi testat din acest lucru (exerciţii ca la seminar).