Tema 4 - Heap-uri. Cozi de prioritate

1. Implementaţi o coadă de priorităţi folosind o structură PRIOROTY_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 (0.5 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)

   Structura trebuie să dispună de constructor. (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. (1 p)

2. Implementarea unui heap d-ar cu aceleaşi operaţii ca pentru heap-ul binar. Un heap d-ar este un arbore d- ar cu proprietăţile de heap cunoscute. (2p)

   Observaţie: Această problemă se poate rezolva prin adaptarea programului de la pb. 1.

3. 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. Utilizanţi o coadă de prioritate (min). (5p)

4. Se consideră o hartă reprezentată sub forma unui graf neorientat ponderat $G = (N, A)$, în care fiecare nod reprezintă un oraş, iar muchiile sunt legăturile (de cale ferată) între oraşe. Pentru fiecare oraş se cunosc: coordonatele spaţiale ale oraşului şi numele acestuia. Pentru fiecare legătură între oraşe se cunoaşte

preţul biletului de tren. Să se realizeze un program în care se vor citi de la tastatură două oraşe A şi B şi se va găsi cel mai ieftin drum de la A la B. Afişaţi traseul şi costul acestiua. Pentru rezolvare folosiţi Dijkstra (5p) sau A* (6p). Utilizaţi o coadă de priorităţi.

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

   Observaţii:

   (a) Rezolvarea problemelor 3, 4 şi 5 presupun utilizarea cozii de prioritate construită la problema 1, dar pe baza unui heap-min. Nu se vor cumula punctele cu punctele de la 1. Acestea sunt incluse în punctajul problemelor.

   (b) Pentru una dintre problemele (3, 4, 5) trebuie implementată complet coada de priorităţi. Pentru celelalte trebuie fi utilizată cea implementată în STL. (în cazul în care va apucaţi de aceste probleme).

   (c) Pentru Dijstra şi A* vă recomand următorul site: https://www.redblobgames.com/pathfinding/a-star/introduction.html