

Tema 4 - Heap-uri. Cozi de prioritate

1. Implementați o coadă de priorități folosind o structură `PRIORITY_QUEUE`, care să aibă un câmp `DATA` de tip vector de întregi, care să stocheze elementele cozii sub forma unui heap max și 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` - crește prioritatea unui nod. (0.5 p)
- `MAX_HEAPIFY` (sau `SIFT_DOWN`) - funcția care coboară o cheie pe poziția corespunzătoare din heap. (0.5 p)

Structura trebuie să dispună de constructor. (0.5 p). În funcția *main* se declară o variabilă de tip `PRIORITY_QUEUE` și se folosește un *menu* implementat cu ajutorul unei instrucțiuni *switch*, prin care utilizatorul să poată selecta oricare dintre operațiile de inserție, extragerea maximului, obținerea maximului și afișarea elementelor din heap. (1 p)

2. Implementarea unui heap d-ar cu aceleași operații ca pentru heap-ul binar. Un heap d-ar este un arbore d- ar cu proprietățile de heap cunoscute. (2p)

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

3. Se citește un text dintr-un fișier. Să se construiască arborele de codificare Huffman corespunzător. Să se afișeze codul corespunzător fiecărui caracter și să se codifica textul. Utilizând 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 oraș, iar muchiile sunt legăturile (de cale ferată) între orașe. Pentru fiecare oraș se cunosc: coordonatele spațiale ale orașului și numele acestuia. Pentru fiecare legătură între orașe se cunoaște

prețul biletului de tren. Să se realizeze un program în care se vor citi de la tastatură două orașe A și B și se va găsi cel mai ieftin drum de la A la B . Afișați traseul și costul acestuia. Pentru rezolvare folosiți Dijkstra (5p) sau A^* (6p). Utilizați o coadă de priorități.

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

Observație: Rezolvarea problemelor cu Dijkstra și A^* (adică 3, 4 și 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.