Uniwersytet Jagielloński Instytut Informatyki Matematyki Komputerowej KIS Katedra Informatyki Stosowanej	Metody programowania 2022/2023 Efektywna implementacja kolejki priorytetowej	P_10

Opis:

Zaimplementuj kolejkę priorytetową wykorzystując tablicową reprezentację max-kopca. Elementami kolejki są liczby całkowite (Int), które mogą występować wielokrotnie. Kolejka ma dwa typy pojemności, N – maksymalna liczbę różnych elementów w kolejce oraz P – maksymalna łączna liczba wszystkich elementów w kolejce.

Priorytet w kolejce jest ustalany następująco:

- jeśli liczba powtórzeń elementu *x* w kolejce jest większa niż liczba powtórzeń elementu *y*, to *x* ma priorytet wyższy niż y.
- jeśli liczby *x* i *y* mają tyle samo wystąpień w kolejce, to większa liczba ma wyższy priorytet.

Wejście/wyjście:

Dane do programu wczytywane są ze standardowego wejścia (klawiatury) zgodnie z poniższą specyfikacją.

1. Pierwszą wartością jest liczba naturalna określająca ilość zestawów danych.
2. Dla każdego zestawu podawane są dwie liczby naturalne wyznaczające pojemności kolejki: N i P dla danego zestawu.
3. W każdym zestawie mogą występować następujące polecenia:
   a. i k x_1 x_2 … x_k – litera *i* sygnalizuję operację *insert*, po niej następuje dodatnia liczba *k* oznaczająca liczbę wstawianych elementów do kolejki oraz same liczby. Jeśli element spowodowałoby przekroczenie jednego z typów pojemności, to zostaje pominięty i nie jest wstawiany do kolejki.

   b. g k – litera *g* sygnalizuję operację *getMax*, usunięcia z kolejki *k* elementów o aktualnie najwyższym priorytecie. Jeśli *k* jest większe od liczby wystąpień elementu o najwyższym priorytecie to usuniętych zostanie tylko liczba tych wystąpień.
   Po usunięciu program powinien wypisać element o najwyższym priorytecie oraz faktyczną liczbę usuniętych powtórzeń, np. załóżmy, że liczba *4* występuje 10-krotnie i posiada aktualnie najwyższy priorytet w kolejce, wtedy polecenie g 3 zmniejszy liczbę wystąpień liczby *4* do z 10-ciu do 7-miu (program wypisze: 4 3), natomiast polecenie g 13 usunie wszystkie wystąpienia liczby *4* (program wypisze: 4 10).
   W przypadku pustej kolejki program powinien wypisać: 0 0

   c. s – litera *s* sygnalizuje operację *sort*, należy wykonać sortowanie kolejki przez kopcowanie (heapsort) i wypisać elementy kolejki rosnąco, zgodnie z priorytetem w formacie: x_1 n_1 x_2 n_2 … x_l, n_l, gdzie x_i to wartość, a n_i to liczba wystąpień i-tego elementu. Złożoność pamięciowa sortowania powinna być O(1), operacja sortowania będzie

zawsze ostatnim poleceniem w zestawie, więc można powyższą złożoność uzyskać kosztem ‘kopcowej’ implementacji kolejki.

Wymagania implementacyjne:

Jak w Programie 1.

Przykład danych:

Wejście	Wyjście
3 4 10 i 5 1 1 1 2 3 g 3 s 5 15 g 2 i 2 1 1 g 1 i 3 2 3 4 g 2 i 7 4 4 4 4 4 1 3 g 3 i 9 2 6 1 6 7 7 7 7 2 g 2 s 8 15 i 10 1 2 3 4 5 6 7 8 9 10 g 2 g 1 g 5 i 10 1 1 1 1 1 1 1 4 4 g 3 i 15 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 g 30 g 2 i 11 3 3 3 5 5 5 5 10 10 2 1 s	1 3 2 1 3 1 0 0 1 1 4 1 4 3 2 2 2 1 3 2 4 2 6 2 1 3 8 1 7 1 6 1 1 3 1 9 4 2 1 1 4 1 2 2 10 2 3 4 5 5