Algorytmy i SD

Struktury danych - Kolejka

Piotr Ciskowski, Łukasz Jeleń
Wrocław, 2023

ADT **kolejka:**

przechowuje dowolne obiekty

- dodawanie i usuwanie – **FIFO**: *First In First Out*
- elementy dodawane są na końcu kolejki, a usuwane z przodu

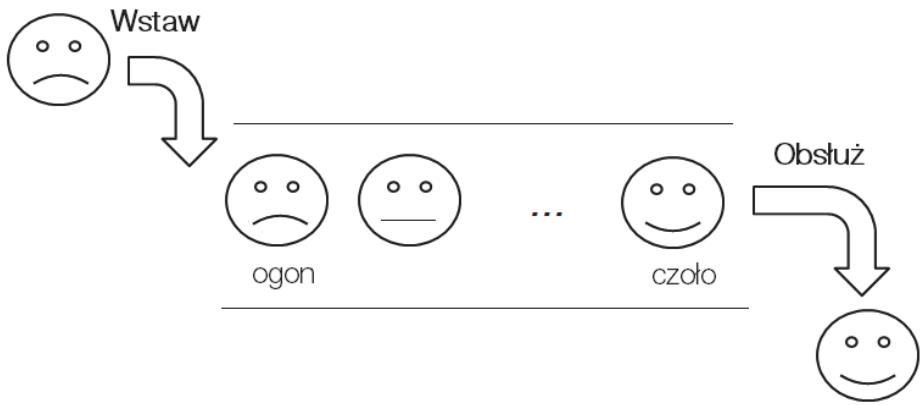
- operacje podstawowe:

  - enque(element) – dodanie elementu na końcu kolejki
  - element dequeue() – usunięcie i zwrócenie elementu z początku

- operacje dodatkowe:

  - element front() – zwraca element na przodzie kolejki bez jego usuwania
  - integer size() – podaje liczbę przechowywanych elementów
  - boolean isEmpty() – mówi, czy na stosie są przechowywane jakieś elementy

struktury danych - kolejka



ADT **kolejka:**

- wyjątki - błędy:
  - front() i dequeue() – próba ich wywołania na pustej kolejce wyrzuci wyjątek *EmptyQueueException*

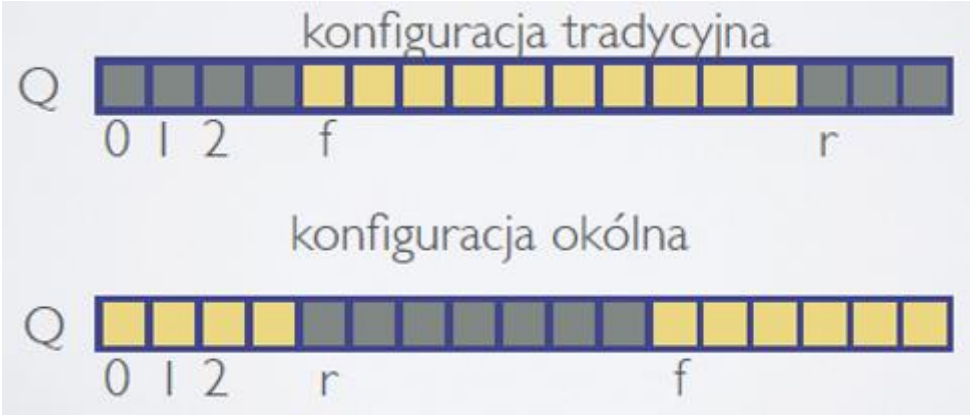

PRZYKŁAD KOLEJKI
Operacja
Wyjście
Q
enqueue(5) – (5)
enqueue(3) – (5, 3)
dequeue() 5 (3)
enqueue(7) – (3, 7)
dequeue() 3 (7)
front() 7 (7)
dequeue() 7 ()
dequeue() "error" ()
isEmpty() true()
enqueue(9) – (9)
enqueue(7) – (9, 7)
size() 2 (9, 7)
enqueue(3) – (9, 7, 3)
enqueue(5) – (9, 7, 3, 5)
dequeue() 9 (7, 3, 5)
© 2004 Goodrich, Tamassia

ADT **kolejka:**

- zastosowania:
  - bezpośrednie – listy oczekujących
    – dostęp do zasobów współdzielonych (drukarka, itp.)
  - pośrednie – struktura pomocnicza dla algorytmów
    – składowa innych struktur danych

kolejka bazująca na tablicy**:**

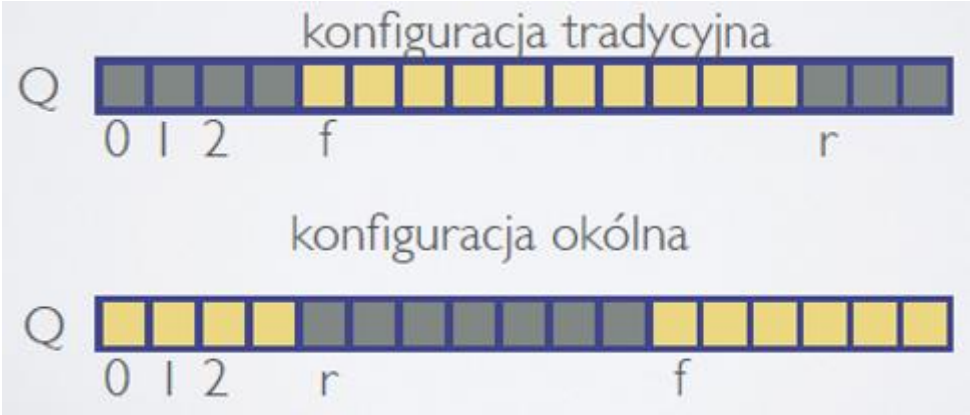
- tablica o rozmiarze *N*
- wykorzystana w sposób tradycyjny – mocno ograniczony lub okrężny/kolisty
- dwie zmienne kontrolując przód i tył kolejki
  - *f* – indeks pierwszego elementu
  - *r* – indeks następny do ostatniego
- pozycja *r* jest pusta



kolejka bazująca na tablicy**:**

- operacje na kolejce
- wykorzystują operator modulo

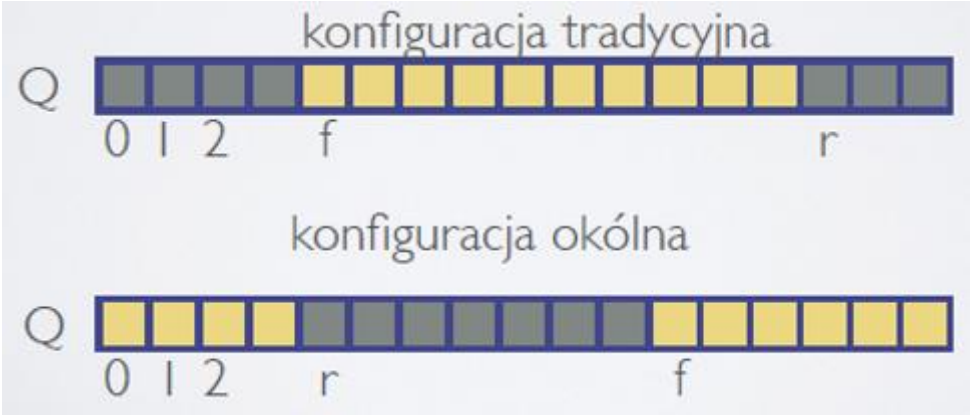
```
Algorytm size()
  return (N − f + r) mod N

Algorytm isEmpty()
  return (f = r)
```



kolejka bazująca na tablicy**:**

- operacja enqueue
  wyrzuca wyjątek
  gdy kolejka jest pełna

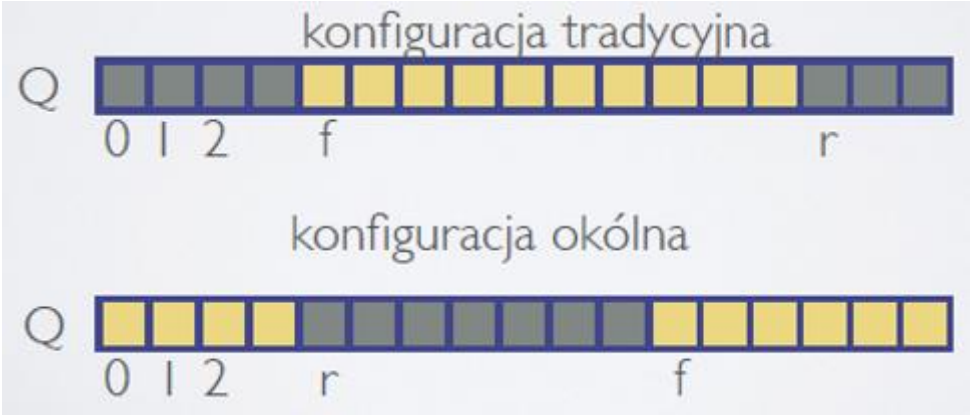
```
Algorytm enqueue(e)
  if size() = N − 1 then
    throw FullQueueException
  else
    Q[r] ← e
    r ← (r + 1) mod N
```



kolejka bazująca na tablicy**:**

- operacja dequeue
  wyrzuca wyjątek
  gdy kolejka jest pusta

```
Algorytm dequeue()
  if isEmpty() then
    throw EmptyQueueException
  else
    temp ← Q[f]
    f ← (f + 1) mod N
    return temp
```



kolejka bazująca na powiększanej tablicy**:**

- analogicznie, jak przy stosie
- gdy tablica się zapełni – można najpierw spróbować znaleźć pamięć na większą tablicę

- średni czas działania operacji enqueue()
  - strategia inkrementalna: $O(n)$
  - strategia podwajania: $O(1)$

nieformalny interfejs w C++

```
template <typename Object>
class Queue{
public:
    int size();
    bool isEmpty();
    Object& front()
          throw(EmptyQueueException);
    void enqueue(Object o);
    Object dequeue()
          throw(EmptyQueueException);
}
```