

Redovi

© Goodrich, Tamassia, Goldwasser

Katedra za informatiku, Fakultet tehničkih nauka, Univerzitet u Novom Sadu

2021.

Red ATP

- **red** (queue) ATP čuva proizvoljne objekte
- ubacivanje i uklanjanje poštuje FIFO (first-in-first-out) princip
- ubacivanje se vrši na kraju reda, a uklanjanje na početku reda
- glavne operacije:
 - **enqueue**(object): dodaje element na kraj reda
 - object **dequeue**(): uklanja element sa početka reda
- dodatne operacije:
 - object **first**(): vraća element sa početka bez uklanjanja
 - integer **len**(): vraća broj elemenata u redu
 - boolean **is_empty**(): vraća True ako je red prazan
- greške:
 - poziv **dequeue** ili **first** za prazan red

Primer operacija nad redom

operacija	rezultat	sadržaj reda
<code>Q.enqueue(5)</code>	–	[5]
<code>Q.enqueue(3)</code>	–	[5,3]
<code>len(Q)</code>	2	[5,3]
<code>Q.dequeue()</code>	5	[3]
<code>Q.is_empty()</code>	False	[3]
<code>Q.dequeue()</code>	3	[]
<code>Q.is_empty()</code>	True	[]
<code>Q.dequeue()</code>	greška	[]
<code>Q.enqueue(7)</code>	–	[7]
<code>Q.enqueue(9)</code>	–	[7,9]
<code>Q.first()</code>	7	[7,9]
<code>Q.enqueue(4)</code>	–	[7,9,4]
<code>len(Q)</code>	3	[7,9,4]
<code>Q.dequeue()</code>	7	[9,4]

Primene redova

- neposredne primene
 - liste čekanja, birokratija
 - pristup deljenim resursima (npr. štampač)
 - deljenje procesora među paralelnim procesima
- indirektne primene
 - pomoćna struktura podataka za mnoge algoritme
 - komponenta u okviru drugih struktura podataka

Red pomoću niza

- implementiraćemo stek pomoću niza dužine N
- koristimo ga cirkularno
- posebne promenljive čuvaju indekse prvog i poslednjeg elementa
 - f – indeks prvog elementa
 - r – indeks prvog elementa iza poslednjeg
- element niza sa indeksom r je uvek prazan!

normalna konfiguracija



premotana konfiguracija



Red pomoću niza ₂

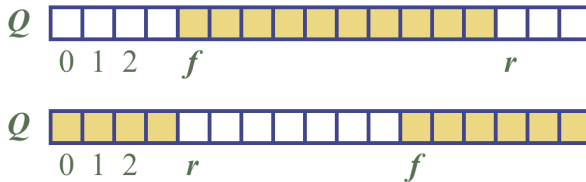
- koristićemo moduo operator (ostatak pri deljenju)
- broj elemenata reda i test da li je prazan:

`size()`

return $(N - f + r) \bmod N$

`isEmpty()`

return $(f = r)$



Red pomoću niza ₃

- operacija **enqueue** izaziva izuzetak ako je niz pun

enqueue(e)

if $\text{size}() = N - 1$ **then**

 throw FullQueueException

else

$Q[r] \leftarrow e$

$r \leftarrow (r + 1) \bmod N$



Red pomoću niza ₄

- operacija **dequeue** izaziva izuzetak ako je red prazan

dequeue()

if isEmpty() **then**

 throw EmptyQueueException

else

$e \leftarrow Q[f]$

$f \leftarrow (f + 1) \bmod N$

return e



Implementacija reda u Pythonu

- imaćemo tri atributa:
- `_data`: lista sa fiksnim kapacitetom
- `_size`: broj elemenata u redu
- `_front`: indeks prvog elementa u redu

Implementacija reda u Pythonu ₁

```
class ArrayQueue:
    DEFAULT_CAPACITY = 10

    def __init__(self):
        self._data = [None] * DEFAULT_CAPACITY
        self._size = 0
        self._front = 0

    def __len__(self):
        return self._size

    def is_empty(self):
        return self._size == 0

    def first(self):
        if self.is_empty():
            raise Empty('queue is empty')
        return self._data[self._front]
```

Implementacija reda u Pythonu 2

```
def dequeue(self):
    if self.is_empty():
        raise Empty('queue is empty')
    answer = self._data[self._front]
    self._data[self._front] = None
    self._front = (self._front + 1) % len(self._data)
    self._size -= 1
    return answer

def enqueue(self, e):
    if self._size == len(self._data):
        self._resize(2*len(self._data))
    avail = (self._front + self._size) % len(self._data)
    self._data[avail] = e
    self._size += 1
```

Implementacija reda u Pythonu 3

```
def _resize(self, cap):  
    old = self._data  
    self._data = [None] * cap  
    walk = self._front  
    for k in range(self._size):  
        self._data[k] = old[walk]  
        walk = (1 + walk) % len(old)  
    self._front = 0
```

Round robin raspoređivanje

- obrada zahteva u krug

1 $e \leftarrow Q.dequeue()$

2 obradi e

3 $Q.enqueue(e)$

