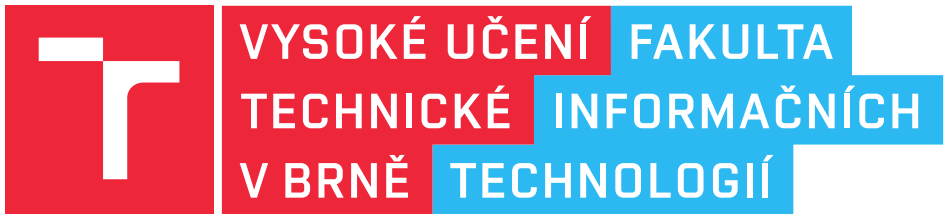
Cyklická fronta

Lukáš Pšeja



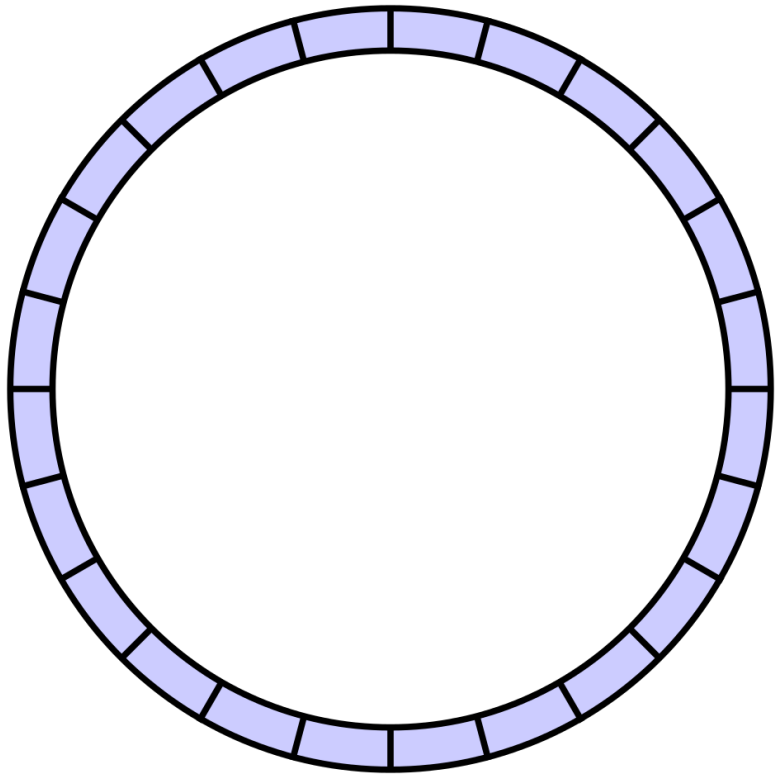
29. dubna 2024

Motivace

- Cyklická fronta se často využívá v praxi
- Je využívána v plánovačích procesů operačních systémů, v síťových systémech při řízení front datových packetů a celkově v systémech, kde je potřeba periodicky zpracovávat události nebo data
- Výhody – rychlost, efektivita paměti, využití pro vyrovnávací paměť
- Nevýhody – omezená velikost

Co je to cyklická fronta?

- Cyklická fronta je datová struktura pro ukládání dat
- Je implementovaná jako zacyklené pole, což umožňuje nekonečný pohyb v poli
- Ke správě slouží dva ukazatele, write a read
- Oproti lineárnímu poli je výhodou, že se nemusí přesouvat prvky, ale pouze posouvají ukazatele



Struktura cyklické fronty

Při implementaci uvažujeme dva způsoby

- Jednodušší způsob, kdy používáme jeden záznam navíc, který nám říká, kolik je momentálně prvků v poli
- Složitější ale efektivnější způsob, kdy používáme ukazatele write a read na výpočet prvků v poli. Aby tato metoda fungovala, musíme ale přidat jeden prvek navíc.

Druhý způsob je výhodnější, protože je vhodný pro multithreading a real-time aplikace.
Jako příklad budeme implementovat UNIXovou funkci tail.

Algoritmus 1: CircularBuffer Struktura

1: **Struktura:** CircularBuffer
2: **Prvky:**
3: • *size* : Integer
4: • *head* : Integer
5: • *tail* : Integer
6: • *lines* : Pointer to Pointer to Character

Kontrola prázdnosti cyklického pole

Algoritmus 2: isEmpty

```
1: Funkce: isEmpty
2: Vstup: CircularBuffer
3: Výstup: Boolean
4: Pokyny:
5:     if head == tail then
6:         return True
7:     else
8:         return False
9:     end if
```

Kontrola plnosti cyklického pole

Algoritmus 3: isFull

```
1: Funkce: isFull
2: Vstup: CircularBuffer
3: Výstup: Boolean
4: Pokyny:
5:     if ((tail + 1) % size) == head; then
6:         return True
7:     else
8:         return False
9:     end if
```

Vložení řádku do cyklického pole

- Časová složitost operace je $O(1)$

Algoritmus 4: cbufPut

```
1:  Funkce: cbufPut
2:  Vstup: CircularBuffer, line
3:  Pokyny:
4:      if isFull; then
5:          free(lines(head))
6:          head = (head + 1) % size
7:      else
8:          lines(tail) = line
9:          tail = (tail + 1) % size
10:     end if
```

Získání řádku z cyklického pole

- Časová složitost operace je $O(1)$

Algoritmus 5: cbufGet

```
1:  Funkce: cbufGet
2:  Vstup: CircularBuffer, index
3:  Výstup: Pointer to Character
4:  Pokyny:
5:      count = (tail - head + size) % size
6:      if index < 0 or index ≥ count; then
7:          return NULL
8:      else
9:          return lines((head + index) % size)
10:     end if
```

Děkuji za pozornost!