# REAL-TIME QUEUE

Tato část bakalářské práce se bude zabývat implementací real-time fronty. Fronta je datová struktura, která je určená pro ukládání a vybírání prvků v takovém pořadí, že prvek, který byl přidán nejdříve bude také nejdříve vybrán. Toto dobře vystihuje anglická zkratka FIFO neboli first in, first out. Pro potřeby implementace byla vytvořena třída Element, která představuje jeden prvek ve frontě.

kod Element

Tato třída obsahuje value tedy hodnotu prvku a také next, což je ukazatel na další prvek.

Existují dvě operace, které lze nad frontou provádět přidání prvku dále značeno jako enqueue a odebrán. Operace dequeue lze provést bez problému v reálném čase. Máme k dispozici ukazatel na první prvek pole, takže pouze přečteme ukazatel next a použijeme tento prvek jako nový začátek fronty. operace enqueue je už horší. Museli bychom projít celou frontu až nakonec, tedy kdy ukazatel next je None a teprv poté napojit nový prvek. To lze ale jednoduše vyřešit tak, že budeme používat místo jednoho seznamu dva.

obrazek headtail

Tyto seznamy budeme podle implementace nazývat head a tail . Head bude reprezentovat první část fronty a bude sloužit pro vybírání prvků z fronty tedy pro operaci dequeue. Tail bude zase reprezentovat druhou část fronty a bude sloužit pro přidávání prvků do fronty tedy pro operaci enqueue. Problém ale může nastat v okamžiku, kdy bychom chtěli odebrat prvek a list head by byl prázdý zatím co tail by prázdný nebyl. V takovém případě bychom museli nejprve přemístit prvky z tail do head a až poté prvek z head odebrat. To by ovšem zabralo n počet operací, kde n je počet prvků v tail. Hlavní myšlenka této implementace je taková, že nikdy nedopustíme aby byl head prázdný, když tail prázdný není. Tohoto bude dosaženo tak, že rozdělíme frontu do stavu. Tyto stavy budou tři (značené od nuly) a budou reprezentovány třídami QueueZero, QueueOne, QueueTwo. QueueZero je používán pro uchování fronty v nultém stavu.

Kod QueueZero

V tomto stavu obsahuje fronta head, tail, tedy dva listy tvořící vlastní frontu, lendiff, který vyjadřuje rozdíl mezi velikostí listu head a velikostí listu tail a bude nám určovat jak dlouho fronta v tomto stavu zůstane. Dále třída QueueZero obsahuje state, který číselně vyjadřuje stav. Vlastnost state bude také součástí dalších tříd vyjadřující ostatní stavy. Operace enqueue v tomto stavu pouze napojí nový Element s požadovanou hodnotou value na tail předchozí fronty a vytvoří novou frontu s head z předchozí fronty nově vytvořeným tail a lendiff předchozí fronty sníženým o jedna. Operace dequeu vytvoří novou frontu přičemž jako head použije ukazatel head.next z předchozí verze fronty, tail zůstane stejný a lendiff předchozí verze snížený o jedna. Tady si můžeme všimnout, že obě operace, jak enqueue tak dequeue snižují hodnotu lendiff. Aby fronta mohla zůstat v nultém stavu, musí být splněna podmínka a to taková, že lendiff > 0. V okamžiku, kdy je tato podmínka porušena přejde fronta do prvního stavu.

V prvním stavu je pro reprezentaci používaná třída QueueOne.

Kod QueueONE

V tomto stavu začneme s přetáčením prvků. při každé operaci nad frontou provedeme dva typy přetáčení

1. Přetáčení prvků z head na head\_reversed
2. Přetáčení prvků z tail na n\_head

Fronta zůstane v prvním stavu dokud jsou v head nebo tail nějaké prvky. Také je dobré zmínit, že problém by nastal kdyby se head (kvůli odebírání prvků) vyprázdnil dříve než je přetáčení dokončeno. To vyřešíme tím, že v každém kroku provedeme přetáčení 1,2 dvakrát, a také provedeme 2 kroky přetáčení při operaci, která zapřičiní přechod fronty ze stavu 0 do stavu 1

Vzorec + dovysvetlit 2n+1

Nové prvky jsou přidávány na n\_tail, ktertý bude v budoucnu tvořit tail fronty a odebírání je řešeno pomocí ukazatele head\_origin, jelikož head bude používán pro přetáčení.

Pokud se kroky přetáčení 1,2 dokončily, fronta přejde ze stavu 1 do stavu 2. Implementací stavu 2 je třída QueueTwo.

Kod QueueTwo

V tomto stavu dochazí k napojení head\_reversed na n\_head vytvořených v předchozím stavu. Hodnota delta\_for\_copy určuje kolik prvků je ještě potřeba napojit. Přidávání prvků je řešeno jednoduše pouze připojením nového prvku na n\_tail. Při odebírání prvků se pouze sníží hodnota delta\_for\_copy a použijeme nový head\_origin přes ukazatel next. Když je proces napojení dokončen, tzn. delta\_for\_copy je rovna 0, fronta přejde zpátky do stavu 0.