Real time deque

V této kapitole, se budeme zabývat implementací deque (double-ended queue), tak aby všechny její operace probíhaly v reálném čase. Toto je typ fronty, u které můžeme operace přidání prvku a odebírání prvku provádět jak zleva tak zprava. Funkce, které řeší tyto operace budeme nazývat pop\_left a pop\_right pro odebírání, a push\_left, push\_right pro přidávání prvků. Dále bude k dispozici funkce new a is\_empty. Funkce new vrátí uživateli novou prázdnou frontu. Funkce is\_empty přijme na vstupu konkrétní frontu a zjistí, zda je tato fronta prázdná nebo ne. Jako základ pro tuto implementaci použijeme pro reprezentaci fronty, stejně jako v předchozí kapitole, dva stacky. Tyto stacky budeme dále značit LHS jako stack levý a RHS jako stack pravý.

Obrazek stacku

Tímto bychom zajistili, že operace pop i push budou prováděný v čase O(1). Problém by ale nastal kdyby se jeden ze stacků operací pop\_left nebo pop\_right vyprázdnil a my chtěli odebírat z daného konce fronty. Pak bychom museli přetočit prvky neprázdného stacku a až potom provést odebírání. Cílem této implementace je rozdělit tento přesun prvků do operací prováděné nad frontou. Proto se bude fronta nacházet v různých stavech a každý tento stav bude realizovaný jednou třídou. Tyto stavy budou čtyři.

# Přetáčení prvků

Nyní se podíváme na samotné přetáčení prvků. Ve stavech 0 a 1 k přetáčení nedochází. Až při přechodu ze stavu 1 do stavu 2 začínají kroky přetáčení. To, že fronta má přejít do stavu 2 poznáme, tak, že bude porušen následující invariant.

Obrazek invariant

Pro vytvoření nových stacků je potřebné vykonat následující kroky:

a.

Přetočení 2m + k – 1 prvků stacku B na pomocný seznam značený aux\_B

b.

Přetočení stacku S na pomocný seznam značený aux\_S

c.

Přetočení zbývající části stacku B na new\_S

d.

Přetočení aux\_B na new\_B

e.

Napojení aux\_S na new\_S

Obrázek přetáčení.

Tyto kroky jsou rozděleny mezi stavy fronty následovně. Ve stavu fronty 2 probíhají kroky a, b.

Ve stavu fronty 3 probíhají zbývající kroky c, d a nakonec také e (s krokem e je možno začít až po dokončení kroku c). Po dokončení posledního kroku tedy napojení aux\_S na new\_S, máme připraveny seznamy new\_S a new\_B, musíme, ale taky počítat s tím, že do fronty budou během přetáčení vkládány a nové prvky. Tyto prvky se budou ukládat do seznamu extra\_S a extra\_B. Jelikož spojení seznamů new\_S a extra\_S, new\_B a extra\_B nelze jednoduše realizovat, použijeme jako reprezentaci stacku dvojici seznamů. V následující části se podíváme na implementaci jednotlivých tříd.

# Element

Prvky fronty budou reprezentovány jako objekty třídy Element.

Kod element

# Stack

Stack je složen ze dvou seznamů tvočených prvky třídy Element, pričemž vždy uchováváme jen ukazatel na první prvek. Tyto ukazatelé jsou v implementaci značeny jako first, pro první část, a second, pro část druhou.

Obrazek stack

Kod stack

Samotný stack má dvě metody:

Metoda **get** vrátí první prvek stacku. Tedy zkontroluje jestli je první část stacku prázdná. Pokud ano vrátí první prvek druhé části tedy ukazatel second. V případě, že první část není prázdná, vrátí jednoduše ukazatel first.

Metoda **next** vrátí novou verzi stacku s odebraným prvním prvkem stacku. Pokud první část není prázdná vezmeme ukazatel first a kontrolujeme, zda za ním existuje další prvek. Pokud ano vrátíme novou verzi stacku pričemž první část bude následující prvek za first tedy first.next a druhá část zůstavá identická jako u předchozí verze. Pokud ovšem další prvek neexistuje, znamená to, že další prvek stacku je až v jeho druhé části. Tedy u nové verze pouze použijeme None pro prvvní část a druhá část zůstane stejná. Nyní případ, kdy první část je prázdná, potom zkontrolujem zda se něco nachází v druhé části. Pokud ne, stack je prázdný a metoda next nedává smysl, vrátíme None. Pokud se nějaký prvek v druhé části nachází vrátíme novou verzi stacku s první částí None a jako ukazatel na druhou část nastavíme následují prvek předchozí verze části second.

# Deque\_0

Tato třída je podobou fronty ve stavu 0. V tomto stavu je fronta reprezentována jako jeden seznam. Fronta může obsahovat v tomto stavu nula až tři prvky.

Kod deque

Operace push zleva realizujeme tak, že vytvoříme nový seznam, vložíme zde požadovaný prvek a poté vložíme všechny prvky z předchozí verze deque. Pokud se bude provádět operace push zprava, budeme postupovat podobně pouze nejprve do nového seznamu vložíme prvky stareé verze až potom nový prvek. Nakonci těchto operací zkontrolujeme délku nového seznamu. Pokud je délka < 4 pouze vrátíme novou verzi fronty ve stavu 0 s nově vytvořeným seznamem. Pokud ovšem délka dosáhne 4, rozpůlíme list na dvě části, z těchto části vytvoříme nový levý stack, pravý stack a vrátíme frontu ve stavu 1.

Kod pushleft

Operace pop zleva a zprava jsou implementované nasledujícím způsobem. Pokud odebíráme zleva vytvoříme nový seznam a z předchozí verze zkopírujeme všechny prvky kromě prvku s indexem 0. Při odebírání zprava zase zkopírujeme do nového seznamu všechny prvky kromš prvku posledního. Toto lze v pythonu krásne zajistit pomoci operátoru [-1]. Jelikož odebíráním nelze překročit maximální velikost fronty v tomto stavvu pouze vrátíme novou verzi fronty ve stavu 0.

Kod push right

# Deque\_1

Tato třída reprezentuje frontu stavu 1.

Kod deque1

Operace push zleva se provádí následovně.Vytvoří se nový LHS a do nšho se na jeho první část *viz Stack*. napojí nový Element s hodnou value, zvětší se délka daného stacku o jedna a poté se zkontroluje nasledujicí invariant

Obrazek invariant

Pokud není porušen vracíme novou verzi fronty ve stavu 1. Pokud ovšem invariant porušen je, dojde k pčechodu do stavu 2. Vytvořím frontu stavu 2 s již přidaným prvkem. Při tomto přechodu je zapotřebí provést na této frontě kroky přetáčení. Obdobně probíhá operace push zprava.

Kod push\_left

Operace pop zleva. Uložíme si hodnotu prvku vybraného z LHS a vytvoříme nový stack zavoláním metody next() na LHS předchozí verze fronty. Snížíme délku daného stacku a zkontrolujeme invariant jak u operace push. Pokud je porušena Druhá podmínka zkontrolujeme nejprve podmínku první. Jestli je porušena první podmínka, znamená to že LHS je prázdný, ale také že v RHS jsou maximálně tři prvky. Vytvoříme tedy z pravého stacku frontu stavu 0 a vrátíme společně s hodnotou vybraného prvku. První podmínka porušena není, fronta přechází do stavu 2 a je potřeba provést počáteční kroky přetáčení obdobně jak u operace push.

Kod pop\_left

# Stav 2

Operace push v tomto stavu napojuji novy prvek na pomocne senznamy extra\_s a extra\_b.

Nejprve zjistime jak je orientovana fronta podle atributu side viz... Napojime prvek na pislusny extra seznam a zvetsime jeho velikost. Provedeme kroky pretaceni a vratime novou verzi fronty ve spravnem stavu.

Operace pop je ponekud slozitejsi. Nejjednodusi pripad je takovy, ze po zjisteni natoceni fronty pres side je prislusny list extra neprazdny. Potom pouze vybereme prvek z tohoto listu vytvorime novou frontu prislusneho stavu a provedeme na ni 4 kroky prislusneho pretaceni. Pokud ale je dany list extra prazdny, musime vybirat prvky z listu origin bude s nebo b podle side a operace.

Potom zjistime hodnotu pomoci aktualniho ukazatele origin nastavime novy ukazatel na nasledujici Element a snizime hodnotu copy\_? Od jedna.

# Stav 3

* Side orientace stacku True znameba S - B False znamena B – S
* S smaller stack and bigger stack B
* S\_origin a B\_origin pro odebirani prvku
* Extra\_s a extra\_b pro pridavani prvku a take delky techto stacku
* Aux counter pocet prvku ktere se z B zkopiruji na aux\_B
* Aux\_s se zkopiruje cely
* Copy\_s copy\_b pocet aktivnich prvku na aux\_s aux\_b
* New\_s, new\_b nove stacky s a b a jejich velikosti S\_size, b\_size

Operace jsou obdobne jak ve stavu 2 pouze s jednou zmenou a to u operaci pop\_left a pop\_right.

Muze totiz nastat pripad takovy. Rekneme ze odebirame zleva nad frontou se side = true tedy usporadani je S --🡪 B. Meli bychom tedy vybirat se seznamu Extra\_s. Ten je ovsem prazdny, potom tedy bychom meli vybirat se seznamu origin\_s. Muze ale nastat situace ze vsechny prvky jsou jiz zkopirovane, tudiz copy\_S je 0, ale fronta je stale ve stavu 3 protoze na druhe strane seznam new\_b jeste neni zcela vytvvoreny (copy\_b > 0). Pak musime postupovat nasledovne. Operaci odebirani provedeme na jiz vytvoreny new\_S. Nova verze bude mit tedy new\_s jako next puvodniho new\_s, ale take nesmime zapomenout snizit velikos new\_s. Obdobne postupujem pri odebirani prvku z vetsiho stacku.