**Žilinská univerzita v Žiline**

Fakulta Riadenia a Informatiky

ANALÝZA VÝKONU ÚDAJOVÝCH ŠTRUKTÚR

Semestrálna práca

Filip Sudora

2020/2021

5ZYI23

**Obsah**

[UML diagram návrhu univerzálnych testov 4](#_Toc68458972)

[Návrh aplikácie a používateľské rozhranie 6](#_Toc68458973)

[Návrh aplikácie 6](#_Toc68458974)

[Používateľské rozhranie 6](#_Toc68458975)

[Popis realizácie priebehu jednotlivých scenárov v testoch 7](#_Toc68458976)

[ADT zoznam 7](#_Toc68458977)

[ADT prioritný front 7](#_Toc68458978)

[ADT viacrozmerné pole – matica 7](#_Toc68458979)

[Údaje zaznamenávané počas testov, formát CSV súboru 8](#_Toc68458980)

[ADT zoznam 8](#_Toc68458981)

[ADT prioritný front 8](#_Toc68458982)

[ADT viacrozmerné pole – matica 8](#_Toc68458983)

[Popis analýzy výsledných dát z testov 9](#_Toc68458984)

[ADT zoznam 9](#_Toc68458985)

[ADT prioritný front 9](#_Toc68458986)

[ADT viacrozmerné pole – matica 9](#_Toc68458987)

[Prezentáciu výsledkov a záverov vyplývajúcich z testovania 10](#_Toc68458988)

[ADT zoznam 10](#_Toc68458989)

[Výsledky ADT zoznam – zoznam implementovaný poľom 10](#_Toc68458990)

[Výsledky ADT zoznam - zreťazená pamäť 12](#_Toc68458991)

[Výsledky ADT zoznam – obojstranne zreťazený cyklický zoznam 14](#_Toc68458992)

[Porovnanie výsledkov ADT zoznam 16](#_Toc68458993)

[ADT prioritný front 17](#_Toc68458994)

[Výsledky ADT prioritný front - zoznam implementovaný poľom, utriedený podľa priorít 17](#_Toc68458995)

[Výsledky ADT prioritný front - ľavostranná halda 19](#_Toc68458996)

[Výsledky ADT prioritný front - Dvoj zoznam ako implementácia prioritného frontu 20](#_Toc68458997)

[Porovnanie výsledkov ADT prioritný front 22](#_Toc68458998)

[ADT viacrozmerné pole – matica 23](#_Toc68458999)

[Výsledky ADT viacrozmerné pole – matica - v súvislej pamäti 23](#_Toc68459000)

[Výsledky ADT viacrozmerné pole – matica - v nesúvislej pamäti 25](#_Toc68459001)

[Porovnanie výsledkov ADT viacrozmerné pole – matica 27](#_Toc68459002)

Zoznam skratiek

AL – Array list – zoznam implementovaný poľom

LL – Linked list – zoznam implementovaný ako zreťazená pamäť

DL – Doubly Circular Linked list - obojstranne zreťazený cyklický zoznam

PQAL – priority queue sorted array list - zoznam implementovaný poľom, utriedený podľa priorít

H – heap – ľavostranná halda

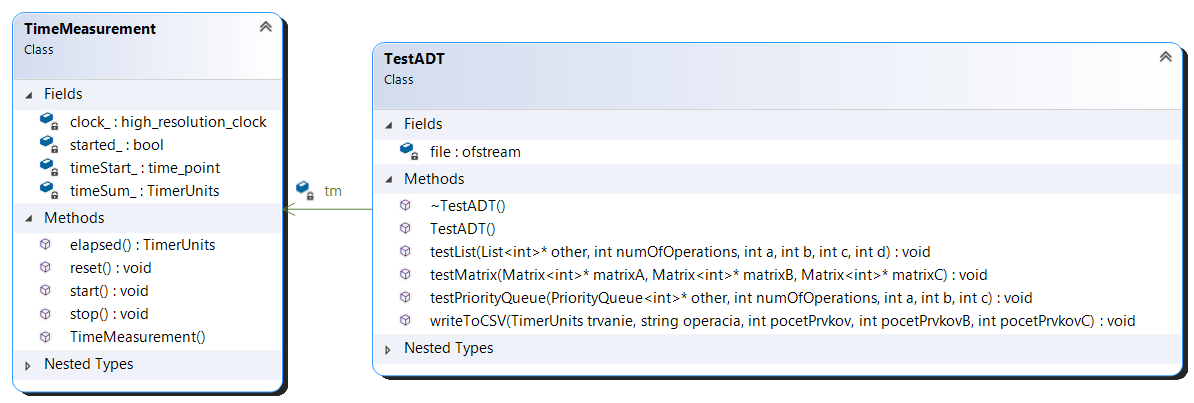
TL – two lists - dvoj zoznam ako implementácia prioritného frontu

C – Contiguous matrix – súvislá matica

NC – Noncontiguous matrix – nesúvislá matica

# UML diagram návrhu univerzálnych testov

Jedna spoločná trieda slúži pre testovanie každej ADT, v atribútoch bude „file“ pre zápis a „tm“ pre meranie času. Do testu budem cez parameter zadávať smerník na danú ADT. Metóda writeToCSV zapisuje údaje do CVS súboru.

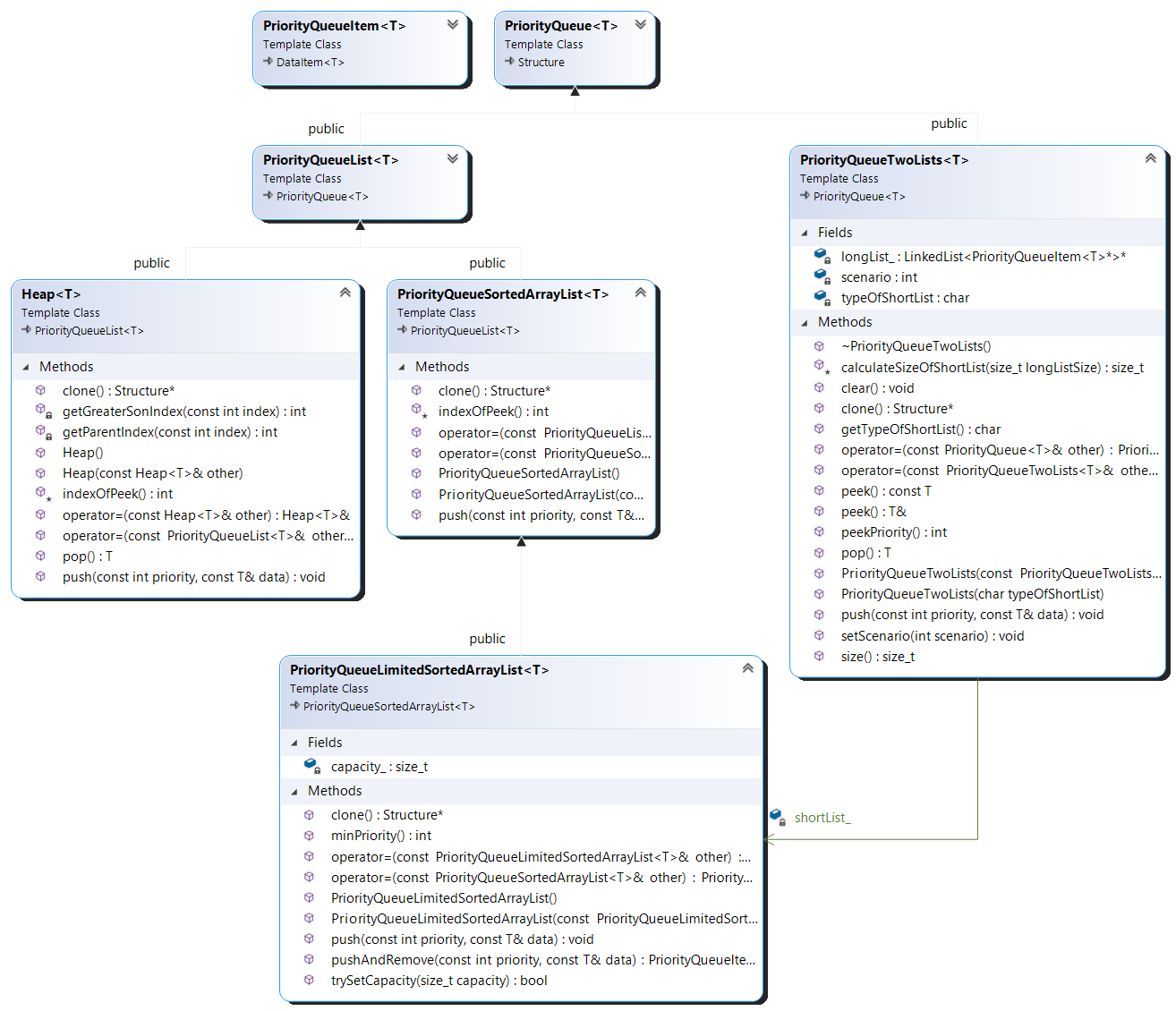


Obrázok - UML diagram triedy univerzálnych testov

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok - UML diagram tried zoznamov



Obrázok - UML diagram tried prioritných frontov

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok - UML diagram tried matíc

# Návrh aplikácie a používateľské rozhranie

## Návrh aplikácie

1. **Main**: Vytvára používateľské rozhranie ovládané cez konzolu a údajové štruktúry.
2. **TestADT**: Obsahuje testy štruktúr a metódu pre zápis do CSV súboru.
3. Ostatné triedy tvoria jednotlivé testované údajové štruktúry.

## Používateľské rozhranie

Po spustení aplikácie sa používateľovi zobrazí konzola pomocou ktorej si bude schopný zvoliť testovanú údajovú štruktúru. V prípade zoznamov a frontov je možnosť nastaviť si vlastný scenár, ten je definovaný počtom operácií a ich podielmi.

Obrázok, na ktorom je text

Automaticky generovaný popisObrázok, na ktorom je text

Automaticky generovaný popisObrázok, na ktorom je text

Automaticky generovaný popis

# Popis realizácie priebehu jednotlivých scenárov v testoch

## ADT zoznam

V **Main** triede si vytvorím inštanciu triedy **ArrayList**, **LinkedList alebo Doubly Circular Linked List**. Následne vytvorený zoznam odošlem to triedy **TestADT** s metódou *testList*, ktorá má ako parameter smerník na predka všetkých zoznamov „List“, kde následne vykonám testované scenáre. Prechádzam postupne všetky scenáre, tie mám uložené v metóde *testList* ako dvojrozmerné pole a vykonávam ich pomocou *for* cyklu. V scenári sa rozhodujem náhodne vygenerovaných číslom, ktoré vyberiem z množiny <0;99>. Následne zisťujem či sa dané číslo nachádza v zadanom rozmedzí, napríklad pre scenár A: <0;19>, <20;39>, <40;89>, <90;99>. Ak vygenerované číslo patrí do intervalu generujem ďalšie číslo s pravdepodobnosť 1 / „počet operácií v skupine“ tým spôsobom, že si vygenerujem národné číslo v rozmedzí 0-2 pre skupinu operácií Vlož, resp. 0-1 pre skupinu operácií Zruš alebo Sprístupni, Nastav. Následne si pred každým zavolaním jednej z operácií zavolám metódu triedy **TimeMeasurement** *start* v ktorej si zapamätám aktuálny čas. Vykonám operáciu, ktorá bola vygenerovaná náhodnými hodnotami. Po skončení operácie zavolám metódu *stop* v ktorej odčítam aktuálny čas od času začiatku vykonávania operácie. Potom prejdem na zápis do súboru pomocou metódy *writeToCSV*.

## ADT prioritný front

V **Main** triede si vytvorím inštanciu triedy **PriorityQueueSortedArrayList**, **Heap** alebo **PriorityQueueTwoLists**. Následne vytvorený prioritný front odošlem to triedy **TestADT** s metódou *testPriorityQueue*, ktorá má ako parameter smerník na predka všetkých prioritných frontov „PriorityQueue“, kde následne vykonám testované scenáre. Prechádzam postupne všetky scenáre, tie mám uložené v metóde *testPriorityQueue* ako dvojrozmerné pole a vykonávam ich pomocou *for* cyklu. V scenári sa rozhodujem náhodne vygenerovaných číslom, ktoré vyberiem z množiny <0;99>. Následne zisťujem či sa dané číslo nachádza v zadanom rozmedzí, napríklad pre scenár A => <0;35>, <36;70>, <71;99>. Následne si pred každým zavolaním jednej z operácií zavolám metódu triedy **TimeMeasurement** *start* v ktorej si zapamätám aktuálny čas. Vykonám operáciu, ktorá bola vygenerovaná náhodnou hodnotou. Po skončení operácie zavolám metódu *stop* v ktorej odčítam aktuálny čas od času začiatku vykonávania operácie. Potom prejdem na zápis do súboru pomocou metódy *writeToCSV*.

## ADT viacrozmerné pole – matica

V **Main** triede si vytvorím 3 inštancie tej istej triedy buď **Contiguous\_Matrix** alebo **Noncontiguous\_Matrix**. Následne vytvorené matice odošlem to triedy **TestADT** s metódou *testMatrix*, ktorá má ako parametre smerníky na predka všetkých matíc „Matrix“, kde následne vykonám testované scenáre. V scenári A/B si urobím vnorený *for* cyklus. Vonkajší *for* cyklus nastavuje dĺžky riadkov matíc a vnútorný cyklus nastavuje dĺžky stĺpcov matíc. Keďže by časová náročnosť sčitovanie/násobenie matíc od 1 po 2000 resp. od 1 po 1500 bola príliš vysoká menil som rozmery matíc nie po 1 ale po 50. Následne si pred každým zavolaním jednej z operácií zavolám metódu triedy **TimeMeasurement** *start* v ktorej si zapamätám aktuálny čas. Vykonám operáciu cez maticový súčet. Po skončení operácie zavolám metódu *stop* v ktorej odčítam aktuálny čas od času začiatku vykonávania operácie. Potom prejdem na zápis do súboru pomocou metódy *writeToCSV.*

# Údaje zaznamenávané počas testov, formát CSV súboru

## ADT zoznam

Počas testov si zaznamenávam po každej vykonanej operácii názov operácie, čas v nanosekundách a aktuálny počet prvkov v zozname. Do CSV súboru zapisujem v nasledovnom formáte. V prvom stĺpci sa nachádzajú názvy operácií, v druhom časy operácií a v treťom aktuálne počty prvkov v zozname. Jeden riadok CSV súboru je napr. vloz prvy;4500;123. Všetky tri vykonané scenáre sú pod sebou.

## ADT prioritný front

Počas testov si zaznamenávam po každej vykonanej operácii názov operácie, napr. „Vlož“, čas v nanosekundách a aktuálny počet prvkov v prioritnom fronte. Do CSV súboru zapisujem v nasledovnom formáte. V prvom stĺpci sa nachádzajú názvy operácií, v druhom časy operácií a v treťom aktuálne počty prvkov v zozname. Jeden riadok CSV súboru je napr. vloz;4500;123. Všetky tri vykonané scenáre sú pod sebou.

## ADT viacrozmerné pole – matica

Počas testov si zaznamenávam po každej vykonanej operácii názov operácie, čas v nanosekundách, počet riadkov výslednej matice, počet stĺpcov výslednej matice. Pre dáta ktoré sú výsledkom maticového násobenia si ukladám aj veľkosti násobených matíc t.j. počet riadkov prvej matice, počet stĺpcov prvej matice, počet riadkov druhej matice, počet stĺpcov druhej matice. Do CSV súboru zapisujem v nasledovnom formáte. V prvom stĺpci sa nachádzajú názvy operácií, v druhom časy operácií, v treťom počty riadkov výslednej matice, v štvrtom počty stĺpcov výslednej matice a navyše pre maticové násobenie v piatom stĺpci počty riadkov prvej matice, v šiestom počty stĺpcov prvej matice, v siedmom počty riadkov druhej matice, vo ôsmom počty stĺpcov druhej matice. Jeden riadok CSV súboru je napr. nasobenie;4500;50;50;50;100;100;50. Obidva vykonané scenáre sú pod sebou.

# Popis analýzy výsledných dát z testov

## ADT zoznam

Zistím si ako dlho trval súčet jednotlivých operácií vzhľadom na celkový čas vykonávania scenáru. Z toho vyvodím ako veľký vplyv má jednotlivá operácia na celkový čas trvania testov. Následne porovnám časovú zložitosť každej operácie od počtu prvkov. Vložím údaje do grafu a tie porovnám s teoretickou časovou zložitosťou. V neposlednom rade porovnám všetky zoznamy a ich priemerné časy operácií. Z týchto dát budem vidieť, ktorý zoznam je pre danú operáciu najvhodnejší.

## ADT prioritný front

Z každého scenára si zistím ako dlho trval súčet operácie vzhľadom na celkový čas vykonávania scenáru. To porovnám s podielmi operácií vstupov do testov. Následne porovnám časovú zložitosť každej operácie od počtu prvkov. Vložím údaje do grafu a tie porovnám s teoretickou časovou zložitosťou. V neposlednom rade porovnám všetky prioritné fronty a ich priemerné časy operácií. Z týchto dát budem vidieť, ktorý prioritný front je pre danú operáciu najvhodnejší.

## ADT viacrozmerné pole – matica

Z nazbieraných dát vytvorím jednu tabuľku pre sčitovanie a jednu pre násobenie, do ktorých prepíšem dáta pre vyhodnocovanie a lepšiu priehľadnosť. Následne porovnám časovú zložitosť oboch operácií od počtu prvkov. Vložím údaje do grafu a tie porovnám s teoretickou časovou zložitosťou. V neposlednom rade porovnám obidve matice a ich priemerné časy operácií. Z týchto dát budem vidieť, ktorá matica je pre danú operáciu vhodnejšia.

# Prezentáciu výsledkov a záverov vyplývajúcich z testovania

## ADT zoznam

### Výsledky ADT zoznam – zoznam implementovaný poľom

Na nasledovnom grafe môžeme vidieť ako vysoko časovo náročné sú jednotlivé operácie. Každej skupine operácií som nastavil podiel 25%. Z grafu jednoznačne vyplýva, že operácia Index prvku je nesmierne časovo zložitá oproti ostatným.

Ako je z grafu možné vidieť, testovaním sme dostali časovú zložitosť operácie Vlož posledný ako konštantnú čo sa zhoduje s teoretickým úsudkom, operácie Vlož prvý a Vlož na index majú len mierne stúpajúci trend, toto je zapríčinené tým, že v pamäti sa musí neustále posúvať väčšie množstvo prvkov, avšak zložitosť môžeme aj napriek tomu označiť ako O(1). Čiže sa zhoduje s  teoretickým úsudkom.

Ako je z grafu možné vidieť, testovaním sme dostali časovú zložitosť operácie Zruš posledný ako konštantnú čo sa zhoduje s teoretickým úsudkom, operácie Zruš prvý a Zruš na indexe majú len mierne stúpajúci trend, toto je zapríčinené tým, že v pamäti sa musí neustále posúvať väčšie množstvo prvkov, avšak zložitosť môžeme aj napriek tomu označiť ako O(1). Čiže sa zhoduje s teoretickým úsudkom.

Z grafu je možné vidieť, že testovaním sme dostali časovú zložitosť operácií Nastav a Sprístupni ako konštantnú čo sa zhoduje s teoretickým úsudkom, operácia Index prvku má spočiatku stúpajúci trend, avšak čím viac prvkov, tým je krivka konštantnejšia. Toto je spôsobené tým, že zoznam môže nadobúdať hodnoty len od 0 po 99 tým pádom je veľká pravdepodobnosť že sa hľadaný prvok nachádza na začiatku zoznamu.

### Výsledky ADT zoznam - zreťazená pamäť

Na nasledovnom grafe môžeme vidieť ako vysoko časovo náročné sú jednotlivé operácie. Každej skupine operácií som nastavil podiel 25%. Z grafu jednoznačne vyplýva, že operácie Zruš posledný, Nastav, Sprístupni sú o mnoho náročnejšie v porovnaní s ostatnými. Je to spôsobené tým, že tieto operácie potrebujú nájsť prvok v zozname na špecifickom indexe. A práve táto operácia je veľmi časovo zložitá v zreťazenej pamäti.

Ako je z ďalšieho grafu možné vidieť, testovaním sme dostali časovú zložitosť operácie Vlož prvý a Vlož posledný ako konštantnú čo sa zhoduje s teoretickým úsudkom, operácia Vlož na index má stúpajúci trend, toto je zapríčinené tým, že je potrebné vyhľadať prvok na indexe o 1 menšom ako vkladaný index čiže zložitosť tejto operácie sa zhoduje s teóriou, čo je O(n).

Ako je z ďalšieho grafu možné vidieť, testovaním sme dostali časovú zložitosť operácie Zruš prvý ako konštantnú čo sa zhoduje s teoretickým úsudkom, operácie Zruš posledný a Zruš na indexe majú zložitosť lineárnu, keďže musím zreťazenú pamäť prehľadávať od začiatku.

Z ďalšieho grafu je možné vidieť, že testovaním sme dostali časovú zložitosť operácií Nastav a Sprístupni ako lineárnu čo sa zhoduje s teoretickým úsudkom, keďže musím prehľadávať daný zoznam od začiatku. Operácia Index prvku má spočiatku stúpajúci trend, avšak čím viac prvkov, tým je krivka konštantnejšia. Toto je spôsobené tým, že zoznam môže nadobúdať hodnoty len od 0 po 99 tým pádom je veľká pravdepodobnosť že sa hľadaný prvok nachádza na začiatku zoznamu.

### Výsledky ADT zoznam – obojstranne zreťazený cyklický zoznam

Na nasledovnom grafe môžeme vidieť ako vysoko časovo náročné sú jednotlivé operácie. Každej skupine operácií som nastavil podiel 25%. Z grafu jednoznačne vyplýva, že operácie Vlož na index, Zruš na indexe, Nastav, Sprístupni, Index prvku sú o mnoho náročnejšie v porovnaní s ostatnými. Je to spôsobené tým, že tieto operácie potrebujú nájsť prvok v zozname na špecifickom indexe. A práve táto operácia je veľmi časovo zložitá v zreťazenej pamäti.

Ako je z ďalšieho grafu možné vidieť, testovaním sme dostali časovú zložitosť operácie Vlož prvý a Vlož posledný ako konštantnú čo sa zhoduje s teoretickým úsudkom, operácia Vlož na index má stúpajúci trend, toto je zapríčinené tým, že je potrebné vyhľadať prvok na indexe o 1 menšom alebo väčšom ako vkladaný index čiže zložitosť tejto operácie sa zhoduje s teóriou, čo je O(n).

Ako je z ďalšieho grafu možné vidieť, testovaním sme dostali časovú zložitosť operácie Zruš prvý a Zruš posledný ako konštantnú čo sa zhoduje s teoretickým úsudkom, operácia Zruš na indexe má zložitosť lineárnu, keďže musím zreťazenú pamäť prehľadávať od začiatku.

Z ďalšieho grafu je možné vidieť, že testovaním sme dostali časovú zložitosť operácií Nastav a Sprístupni ako lineárnu čo sa zhoduje s teóriou, keďže musím prehľadávať daný zoznam od začiatku, resp. konca. Operácia Index prvku má spočiatku stúpajúci trend, avšak čím viac prvkov, tým je krivka konštantnejšia. Toto je spôsobené tým, že zoznam môže nadobúdať hodnoty len od 0 po 99, tým pádom je veľká pravdepodobnosť že sa hľadaný prvok nachádza na začiatku zoznamu.

### Porovnanie výsledkov ADT zoznam

Na uvedenom grafe vidíme priemerný čas vykonania každej operácie vo všetkých zoznamoch.

## ADT prioritný front

### Výsledky ADT prioritný front - zoznam implementovaný poľom, utriedený podľa priorít

Ako je z ďalšieho grafu možné vidieť, testovaním sme dostali časovú zložitosť operácie Vlož ako logaritmickú čo sa zhoduje s teoretickým úsudkom.

Ako je z ďalšieho grafu možné vidieť, testovaním sme dostali konštantnú časovú zložitosť operácie Vyber. Táto operácia je konštantná pretože prvok s maximálnou prioritou je vždy na konci.

Z ďalšieho grafu je možné vidieť, že testovaním sme dostali časovú zložitosť operácie Ukáž konštantnú. Táto operácia je konštantná pretože prvok s maximálnou prioritou je vždy na konci.

### Výsledky ADT prioritný front - ľavostranná halda

Na nasledujúcom grafe je možné vidieť, že testovaním sme dostali časovú zložitosť operácie Vlož O(log N). Táto zložitosť sa zhoduje s teoretickým úsudkom.

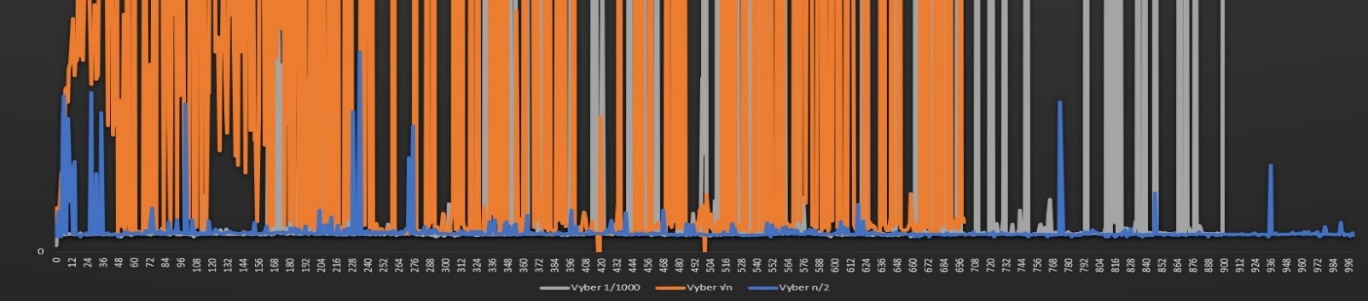
Na nasledujúcom grafe je možné vidieť, že testovaním sme dostali časovú zložitosť operácie Vyber O(log N). Táto zložitosť sa zhoduje s teoretickým predpokladom.

Z ďalšieho grafu vyplýva, že testovaním sme dostali časovú zložitosť operácie Ukáž O(1). Táto zložitosť sa zhoduje s teoretickým predpokladom. Keďže prvok s najväčšou prioritou je v halde ako prvý.

### Výsledky ADT prioritný front - Dvoj zoznam ako implementácia prioritného frontu

Z nasledujúceho grafu vyplýva, že časová zložitosť operácie Vlož je konštantná, pretože väčšinou sa vkladá do dlhšieho zoznamu. Čo súhlasí s teoretickým predpokladom.

Z nasledujúceho grafu je vidieť správanie sa operácie Vyber pri dvoj zozname. Zložitosť tejto operácie je rovnaká ako je dĺžka menšieho usporiadaného zoznamu. Avšak pri reštrukturalizácií sa zložitosť zvýši.



Z ďalšieho grafu vyplýva, že časová zložitosť operácie Ukáž je O(1). Je to dané tým že prvok s najväčšou prioritou je v menšom utriedenom fronte.

### Porovnanie výsledkov ADT prioritný front

Na uvedenom grafe vidíme priemerný čas vykonania každej operácie vo všetkých typoch prioritných frontov.

## ADT viacrozmerné pole – matica

### Výsledky ADT viacrozmerné pole – matica - v súvislej pamäti

Z uvedeného grafu odhadujem, že časová zložitosť sčitovania dvoch matíc v súvislej pamäti je O(n\*m).

Ako môžeme vidieť na grafe nižšie, nezáleží ako meníme riadkový a stĺpcový index pri sčitovaní dvoch matíc.

Z uvedeného grafu odhadujem, že časová zložitosť násobenia dvoch matíc v súvislej pamäti je O(n^3).

Na nasledovnom grafe môžeme vidieť, že menenie počtu riadkov nám viac vplýva na násobenie matíc v súvislej pamäti.

### Výsledky ADT viacrozmerné pole – matica - v nesúvislej pamäti

Z uvedeného grafu odhadujem, že časová zložitosť sčitovania dvoch matíc v nesúvislej pamäti je O(n\*m).

Ako môžeme vidieť na grafe nižšie, nezáleží ako meníme riadkový a stĺpcový index pri sčitovaní dvoch matíc v nesúvislej pamäti.

Z uvedeného grafu odhadujem, že časová zložitosť násobenia dvoch matíc v nesúvislej pamäti je O(n^3).

Na nasledovnom grafe môžeme vidieť, že menenie počtu riadkov nám viac vplýva na násobenie matíc v nesúvislej pamäti.

### Porovnanie výsledkov ADT viacrozmerné pole – matica

Pri sčitovaní aj násobení je efektívnejšie využiť maticu implementovanú v súvislej pamäti.

