**Žilinská Univerzita v Žiline**

**Fakulta Riadenia a Informatiky**

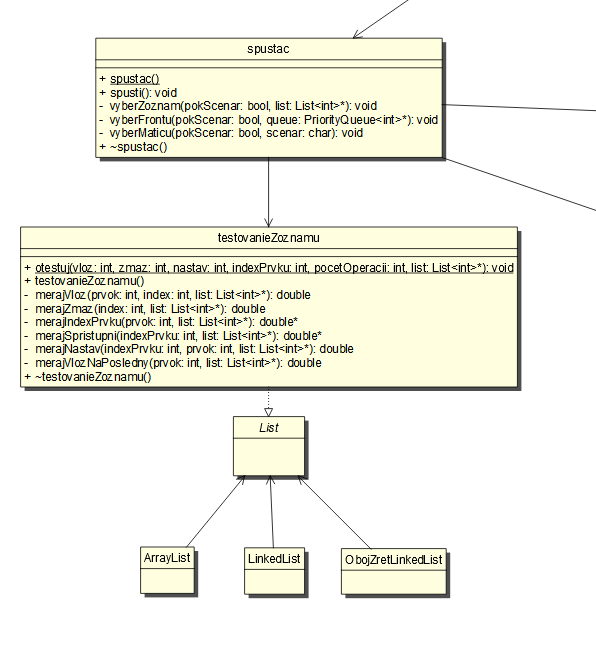
Algoritmy a Údajové Štruktúry 1

1. Semestrálna práca

DOKUMENTÁCIA

**ADT zoznam**

**• Vytvorte UML diagram tried univerzálneho návrhu testov pre ADT zoznam a popíšte ho.**



UML diagram pre zoznamy sa skladá z tried na obrazku. Pomocou týchto tried som skúmal čas operacií podľa scenárov, ktoré si užívateľ aplikácie može vyskladať podľa seba, taktiež si môže vytvoriť svoj vlastný scenár. Testovanie prebieha v metóde testovanieZoznamu, do ktorej sa dostanem z triedy spustac. Tá má na starosti vybrať scenár, ADT štruktúru a druh zoznamu podľa toho čo chce používateľ.

**• Popíšte, ako budete realizovať jednotlivé scenáre pri testovaní ADT zoznam. Teda, ako sa budete v danom scenári rozhodovať, ktorú skupinu operácií zvolíte a ktorú operáciu z danej skupiny následne vykonáte.**

V tejto časti by som rád spomenul to, že je poradie operácii veľmi dôležité, i ked je náhodné. Pokial mám prázdny zoznam nemôžem použiť operáciu zrus, sprístupni, nastav. Pretože jednoducho nemám čo rušiť, vyberať atd. Z toho nám vyplíva, že operácia vlož sa vykaná vždy ako prvá a ostatné operácie sa môžu vykonávať iba vtedy pokiaľ zoznam nie je prázdny. To nám zapríčiní, že zo 100000 operácií budem platných len nejakých 97%. Tento problém som vyriešil tým sposobom, že po každej neplatnej operácii sa vykoná jedna operácia navyše, čo mi dá konečný počet 100000 platných operácií.

**• Popíšte údaje, zaznamenávané počas testov, a formát CSV súboru (teda, čo sa bude v CSV súbore nachádzať a kde), do ktorého budete tieto údaje ukladať.**

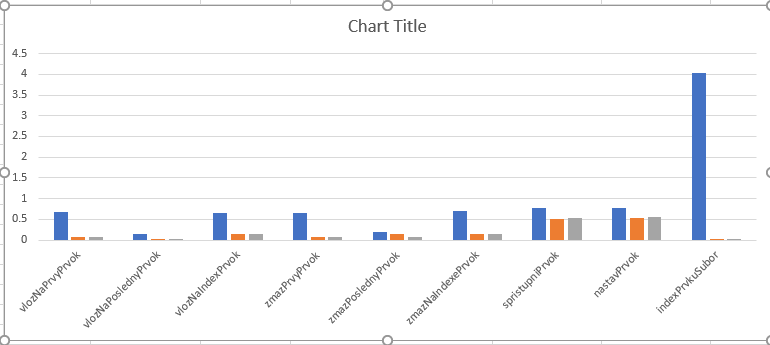
Pre každú operáciu som si vytvoril samostatný súbor, do ktorého sa vygenerujú dáta. Väčšinou ide o čas operácie, dĺžka zoznamu, index prvku s ktorým pracujem a jeho hodnota. Dáta budú oddelené v každom riadku čiarkou. Po kazdej operácii vysledky zapíšem do súboru. Súbor .csv možem otvoriť v Exceli, v ktorom budem analyzovať získané dáta.

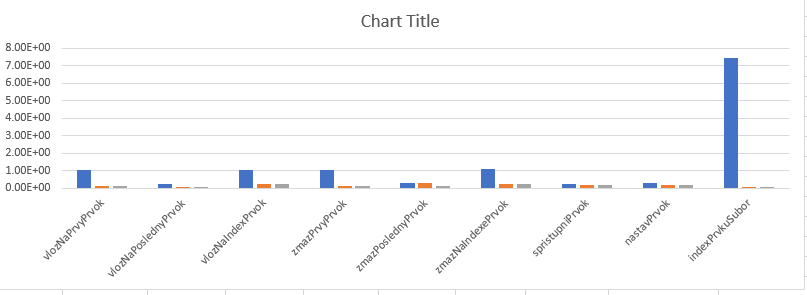
**• Popíšte, ako budete analyzovať výsledné CSV súbory.**

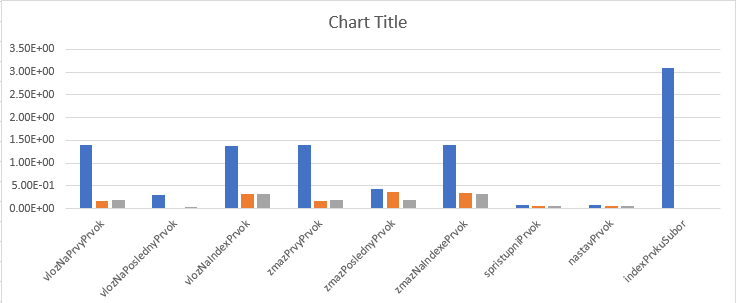
Z CSV súborov sa budú dať vygenerovať grafy v Exceli, pomocou týchto grafov môžem zistiť, ktorý zoznam je vhodnejší na používanie. Tiež si môžem zistiť minimálny čas potrebný na vykonanie operácie, maximálny čas na vykonanie operácie, priemerný čas atď. Toto porovnávanie mi pomôže zistiť efektivitu operácií.

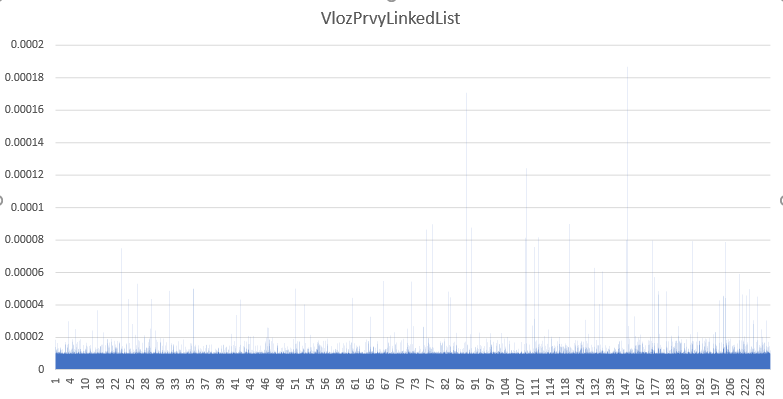


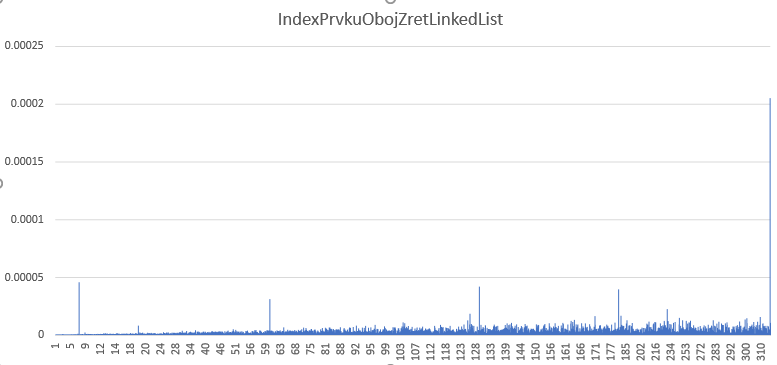
Zistenú zložitosť overíme s teoretickými zložitosťami, ktoré už vieme. Mal by som sa približovať ku teoretickým hodnotám.

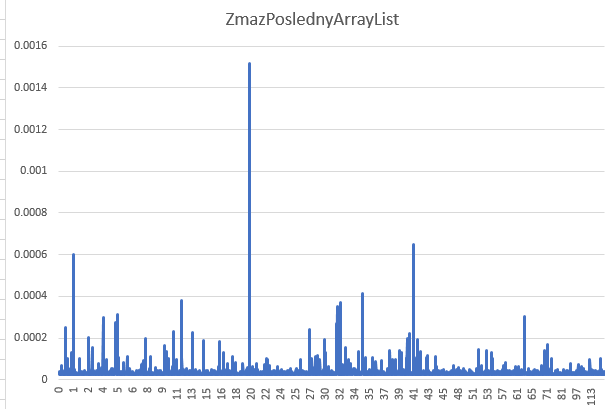
Grafíky









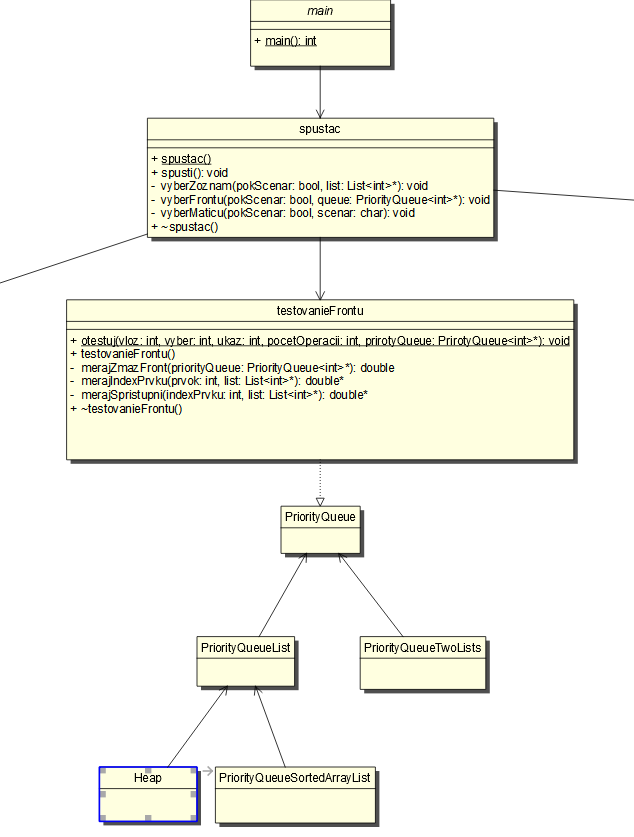


Pre dalšie dáta pozrite súbor ***prehladcasov.xlsx***

**• Zhrnutie:**

Z pozorovania grafov nám vyplýva, že najrýchlejší je Obojstranne zreťazený cyklický zoznam. Správa sa podobne ako Linked List s tým rozdielom, že prehľadávať prvky môže aj od začiatku aj od konca. Toto teoreticky znižuje dlžku vyhladávania až o polovicu. Pokial mame prvok na indexe v strede zoznamu. Najdlhšie trvajú operácie v ArrayListe a to hlavne Index Prvku pretoze prechádzame celé pole, aby sme mohli nájsť náš požadovaný prvok. Vytvorené csv súbory nám dovolili vytvoriť si grafy a porovnať zoznamy. LinkedList je rýchlejší, pretože každý prvok si pamätá svojho nasledovníka. Najdlhšie mu trvá zrušPosledný, kedže musí prejsť cez celý zoznam.

**ADT prioritný front**

**• Vytvorte UML diagram tried univerzálneho návrhu testov pre ADT zoznam a popíšte ho.**

UML diagram pre zoznamy sa skladá z tried na obrazku. Pomocou týchto tried som skúmal čas operacií podľa scenárov, ktoré si užívateľ aplikácie može vyskladať podľa seba, taktiež si môže vytvoriť svoj vlastný scenár. Testovanie prebieha v metóde testovanieZoznamu, do ktorej sa dostanem z triedy spustac. Tá má na starosti vybrať scenár, ADT štruktúru a druh zoznamu podľa toho čo chce používateľ.

**• Popíšte, ako budete realizovať jednotlivé scenáre pri testovaní ADT zoznam. Teda, ako sa budete v danom scenári rozhodovať, ktorú skupinu operácií zvolíte a ktorú operáciu z danej skupiny následne vykonáte.**

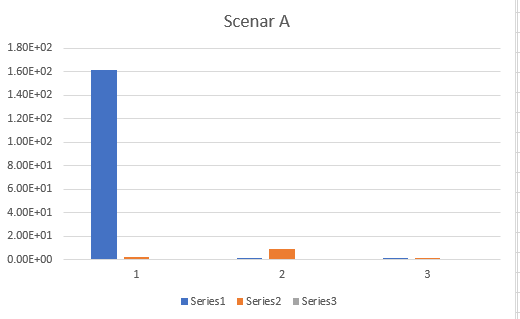
Prioritné fronty testujem pomocou triedy TestovanieFrontu. Táto trieda ma public metódu otestuj, ktorá volá private metódy merajOperáciu. Otestuj je ako keby kostra, v ktorej sa vykoná 100 000 operácií, podľa určítych parametrov. Metódy merajOperáciu vykonajú a odmerajú čas jednej operácie. Tak ako pri zoznamoch je potrebné, aby sa vykonali iba operácie, ktoré sú valídne. Nemôžeme vybrať nič z prázdneho frontu... Toto a dalšie zabezpečuje otestuj.

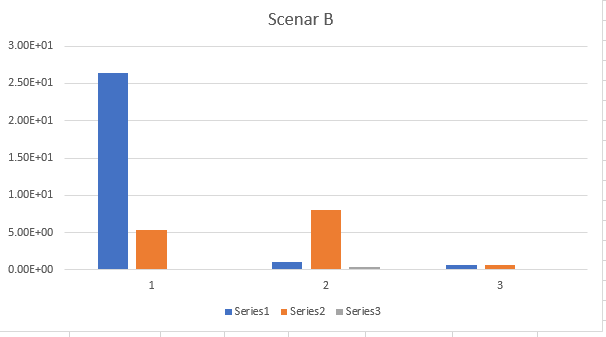
**• Popíšte údaje, zaznamenávané počas testov, a formát CSV súboru (teda, čo sa bude v CSV súbore nachádzať a kde), do ktorého budete tieto údaje ukladať.**

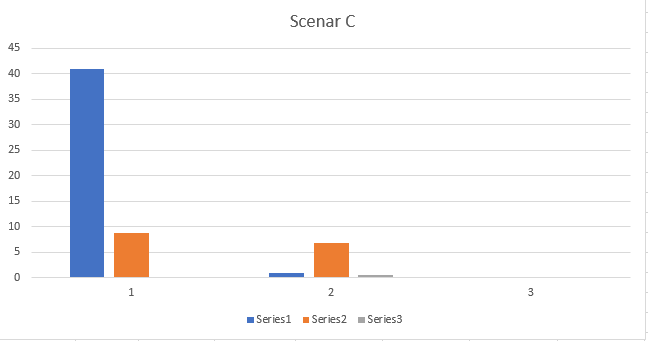
Generovanie sa podoba generovaniu zoznamov. Zmena v týchto dvoch ADT je v priorite. Zoznamy prioritu nemaju, prioritne fronty ano. To znamená, že fronty sa budu deliť na zoradené a nezoradené. Vytvoríme si súbor na meranie každej operácie zapíšeme si čas, dĺžku zoznamu a ku niektorým operáciám aj prioritu. Z dát získame tiež celkový počet operácií, celkový čas jedného druhu operáciea priemerný čas z ktorý sa operácia vykonala.

**• Popíšte, ako budete analyzovať výsledné CSV súbory.**

Z dát, ktoré získame vytvoríme grafy, ktoré nám pomôžu určiť si, ktorý zoznam má aké rýchle spracovanie operácií. Vieme porovnať celkový čas spracovania jedného druhu operácie, premerné časy operácií. Vieme si vytvoriť graf z ćasu operácií a prioritného frontu. Takéto grafy môžete vidieť nižšie. Zvyšok grafov je v ***prehladcasov.xlsx.***







Porovnanie rychlosti spracovania operácií PriorityQueueTwoLists scenar B

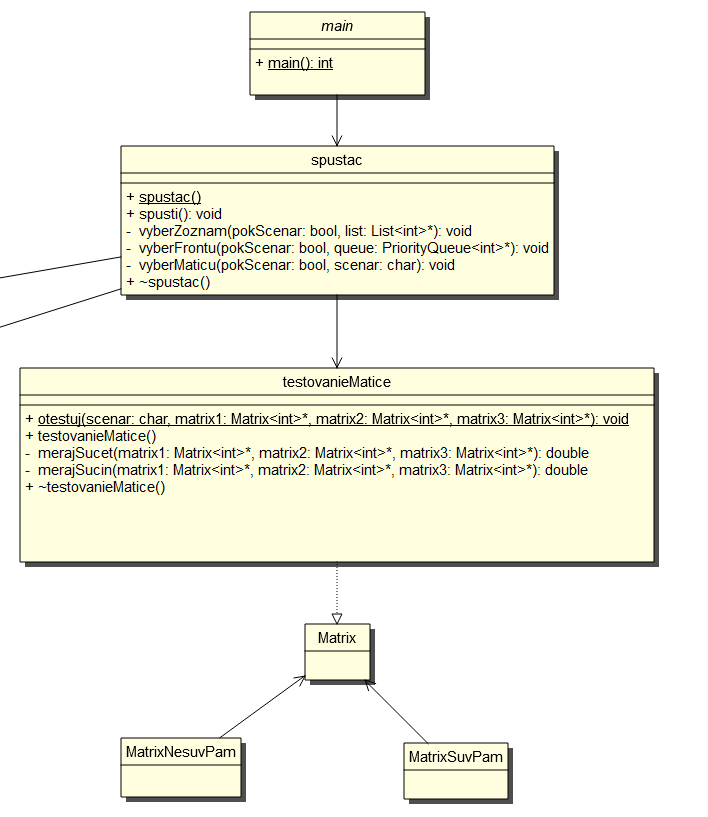
**• Zhrnutie:**

Pri testovani scenara B pre TwoLists som zistil ze najrýchlejšie sa vykonali operácie, keď som si zvolil odmocninu n, takýto výsledok potvrdzuje to, že najefektívnejšia dĺžka ArrayListu je odmocnina z LinkedListu. Za ňou bol zlomok z n/2 a na poslednom mieste je konstanta, ktorej to trvalo najdlhšie.

Co sa týka porovnávania prioritných frontov. Zistil som že TwoLists je najrýchlejšia PQ štruktúra, ktorú sme si testovali, za ňou nasleduje Halda a na koniec PQsortedAL. Toto zoradenie dáva zmysel pretože TwoLists majú teoretickú rýchlost n1\*/2, využívajú kratší utriedený front podľa priorít a dlhší neutriedený front, halda je binárny strom, čo znamená, že neprehľadávame velý strom ale iba jednu vetvu. No a PQsortedAL má ten problem, že pri vkladaní porovnáva priority a tak sa pohybuje po zozname. Čiže čím väčší rozsah priorít tým dlhšie vyhľadávanie trvá.

**ADT viacrozmerné pole – matica**

**• Vytvorte UML diagram tried univerzálneho návrhu testov pre ADT zoznam a popíšte ho**



Na testovanie matice využívam triedu TestovanieMatice. V tejto triede mám na vyber medzi meraním súčtu alebo meraním súčinu. Metódu otestuj volám toľko krát, koľko operácií chcem otestovať. Pre testovanie som si vytvoril dve triedy. Jednu pre súvislú maticu a druhú pre nesúvislú maticu. Tieto dve triedy dedia z predka Matica. Pole v súvislej matici je Array<T>, ktorá má rozmer1 \* rozmer2. Nesúvislá matica má pole polí Array<Array<T>\*>. Do týchto polí sa ukladajú prvky T v našom prípade sú to integery.

**• Popíšte, ako budete realizovať jednotlivé scenáre pri testovaní ADT zoznam. Teda, ako sa budete v danom scenári rozhodovať, ktorú skupinu operácií zvolíte a ktorú operáciu z danej skupiny následne vykonáte.**

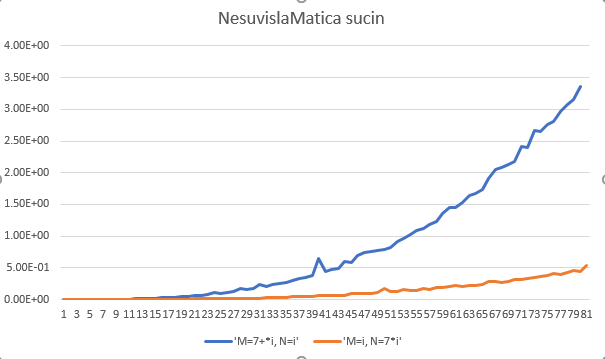
Vytvoril som si cyklus, ktorý sa opakuje 80 krát a zvyšuje parametre matice na grafoch je pekne vidieť, ktorý z paramätrov m a n má väčší vplyv na výkon pri počítaní v matici. Hodnoty sa zväčšujú o 7 v jednom rozmere a o 1 v druhom. Po vykonaní testu sa rozmery vymenia, aby sme mohli porovnať aký časvý rozdiel dostaneme pokiaľ bude 1. rozmer rásť o 1 a 2. rozmer o 7. Toto urobíme aj pre sčítanie aj pre násobenie.

**• Popíšte údaje, zaznamenávané počas testov, a formát CSV súboru (teda, čo sa bude v CSV súbore nachádzať a kde), do ktorého budete tieto údaje ukladať.**

V .csv súbore sa bude nachádzať čas operácie, rozmer1 a rozmer2 toto sú rozmery výslednej matice. Ukladať sa budú do samostatných csv suborov. Je vhodné, aby v jednom dokumente boli údaje pri ktorých sa mení 1. rozmer matice sedem násobne viac a údaje, pri ktorých sa mení 2. rozmer matice 7 násobne viac spolu (lepšie je to vidieť na grafe).

**• Popíšte, ako budete analyzovať výsledné CSV súbory.**

Z údajov, ktoré som popisoval vyššie, zostavím grafy. Tieto grafy porovnávajú vplyv hodnôt m a n na súčet a súčin matíc. Grafy, ktoré vytvorím môžem následne porovnať medzi sebou ( súvislá, nesúvislá matica) a zistím, ktorá matica je rýchlejšia.



**• Zhrnutie:**

Ako môžeme vidieť na vzorke pri scenári A oba rozmery majú viac menej rovnaký vplyv na čas operácie súčet. Pri väčšej vzorke by skoky, ktoré vidíme neboli tak viditelné. Môžeme vidieť, že stúpajú tým istým tempom.

Pri násobení matíc vidíme, že rozmer M má oveľa väčší vplyv na čas než rozmer N. Pri nesúvislej matici stúpol na takmer 3 a pol sekúnd pri matici 80\*7, 80.

Z grafov vplýva, že sa nám viacej oplatí používať súvislú maticu a to hlavne z dovodu, že nám ušetrí čas pri násobení, no taktiež jej rýchlosť sčítania nie je zanedbateľná.