Fakulta riadenia a informatiky
Informatika

Analýza výkonu údajových štruktúr

1. semestrálnej práce

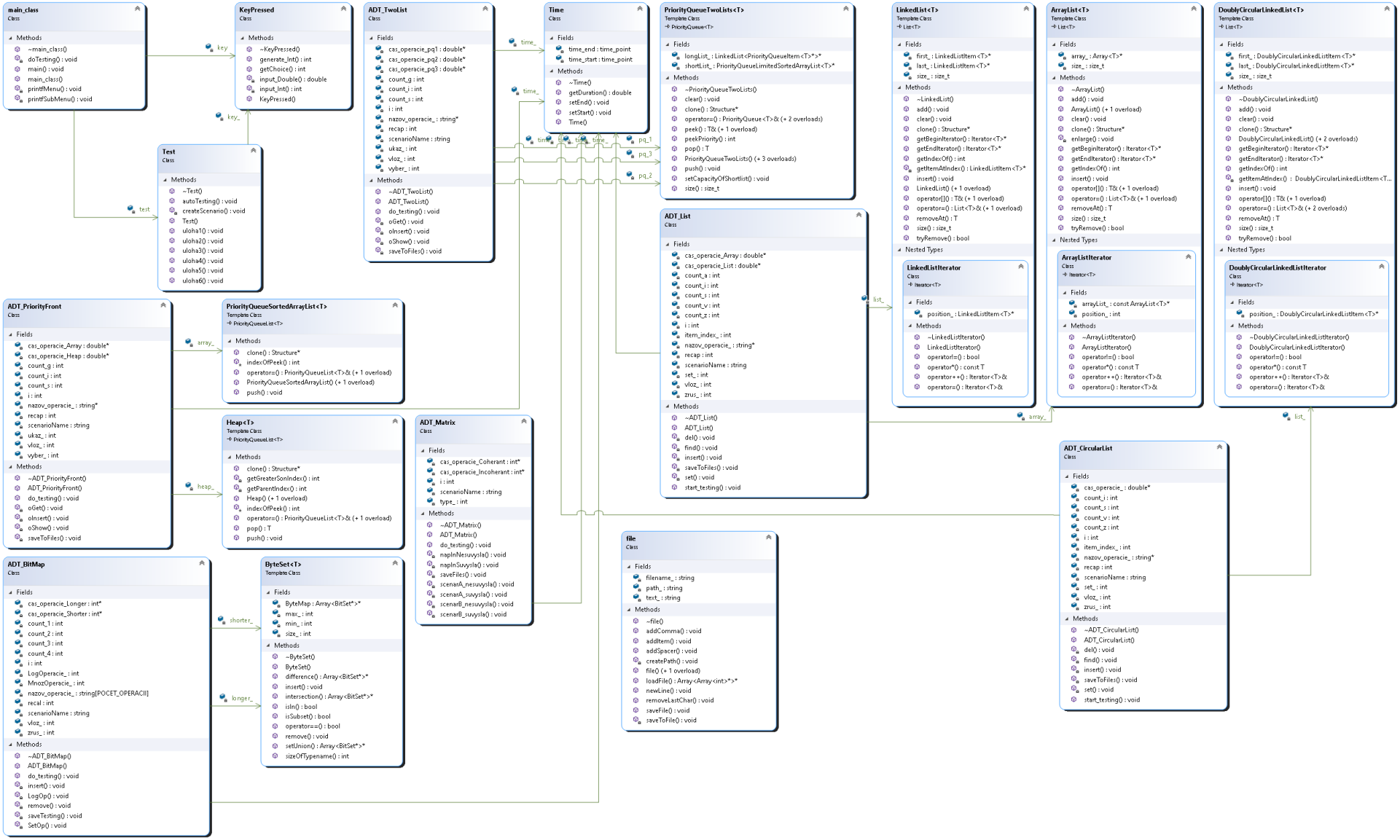
2020/2021 Michal Molitoris, *5ZYI3B*

Obsah

Popis aplikácie

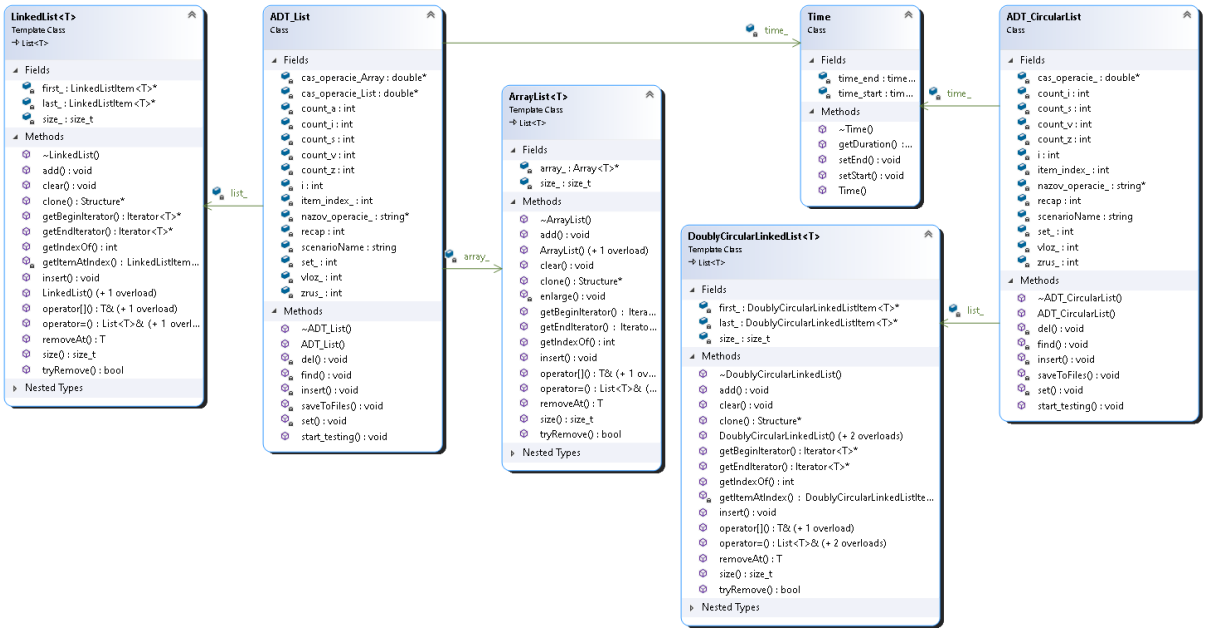
Aplikácia opísaná v tomto dokumente slúži na testovanie vybraných dátových štruktúr. Štruktúry sú testované všeobecne a na všetkých operáciách, ktoré sú schopné dané štruktúry vykonávať. Testovanie štruktúr prebieha buď pomocou pred pripravených scenárov alebo je na výber pre užívateľa možnosť vymyslieť a vytvoriť si vlastné scenáre a nechať ich automatický od testovať pre danú štruktúru, ktorú si užívateľ zvolil.V samotných testoch sú jednotlivé operácie volané automaticky a náhodne a tým pádom sa testy vykonávajú a simulujú normálnu prácu s danou testovacou štruktúrov. Po vykonaného zvoleného testu sa výsledky ukladajú do csv súborov, pomocou ktorých je možné tieto výsledky ľahko analyzovať.

UML diagram aplikácie



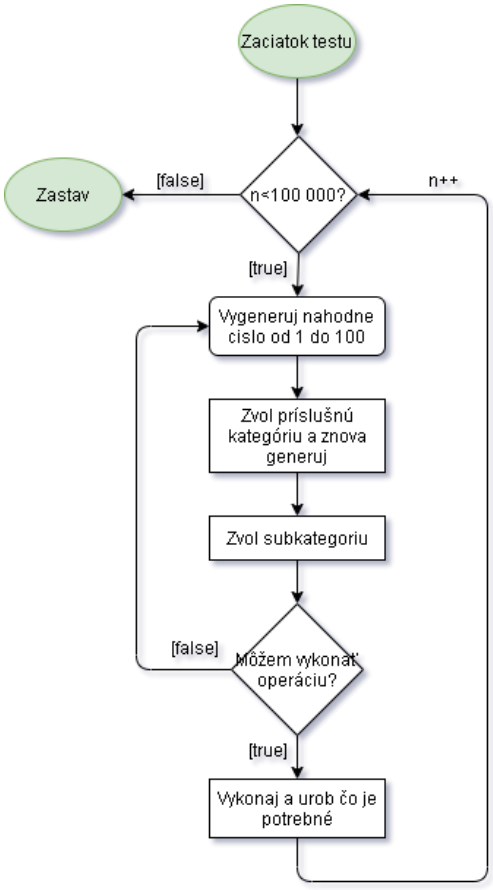
ADT zoznam a Obojstranne zreťazený cyklický zoznam

UML diagram návrhu univerzálnych testov



Po zvolený testovanej dátovej štruktúry, v tomto prípade dátovej štruktúry ArrayList, LinkedList a DoubleCircularLinkedList sa zavolá funkcia z triedy Test ktorá prislúcha daným testom. V tejto funkcii si užívateľ zvolý daný scenár ktorý chce odtestovať alebo si vytvorý vlastný. Po zvolený testu sa Vytvorý trieda ADT_List (alebo ADT_DoublyCircularLinkedList) a zadefinujú sa parameter prislúchajúce danému scenáru a vykoná sa testovanie z ktorého výsledky sa uložia do súboru prostredníctvom triedy File.

Navrh a priebeh testov



Po štarte zvoleného scenáru sa začne cyklus, ktorý predstavuje jednotlivé operácie, ktoré sa počas testovania štruktúr vykonávajú. Pri každej operácií začíname generovaním náhodnej hodnoty, pomocou ktorej zvolíme kategóriu operácie (jednotlivé pravdepodobnosti sú zadané pri spúštaný konkrétneho scenára), obdobným spôsobom zvolíme subkategóriu, ktorá prislúcha konkrétnej operácii a zistíme či je možné túto operáciu vykonať. V prípade, že nie, treba znova voliť kategóriu a neskôr subkategóriu. Rovnako tak treba zaručiť, aby percentuálne sedel počet jednotlivých operácií. A však ak je možné operáciu vykonať postupujeme tým, že zaznamenáme čas začiatku, vykonáme danú operáciu a znova zaznamenáme čas konca. Pomocou času konca a začiatku vieme určiť, ako dlho sa vykonávala operácia. Tieto časy zaznamenávame a zapisujeme ich do súboru, zároveň ich používame na zistenie celkového času.

Keďže pri porovnávaní dvoch a viacerých štruktúr treba zaručiť, aby boli obe testované rovnakými parametrami voláme danú operáciu najskôr pre jeden typ a neskôr pre druhý.

Formát údajov a súboru

Výsledky testu sa budú ukladať do súboru, v ktorom sa budú zaznamenávané jednotlivé typy operácií a časy ich trvania pre obe nami testované dátové štruktúry. Pomocou tohto súboru si následne môžme vypočítať priemery pre každý druh operácie a zároveň celkový čas na vykonanie jednej operácie, pomocou ktorých vyvodíme závery vyplývajúce z testu.

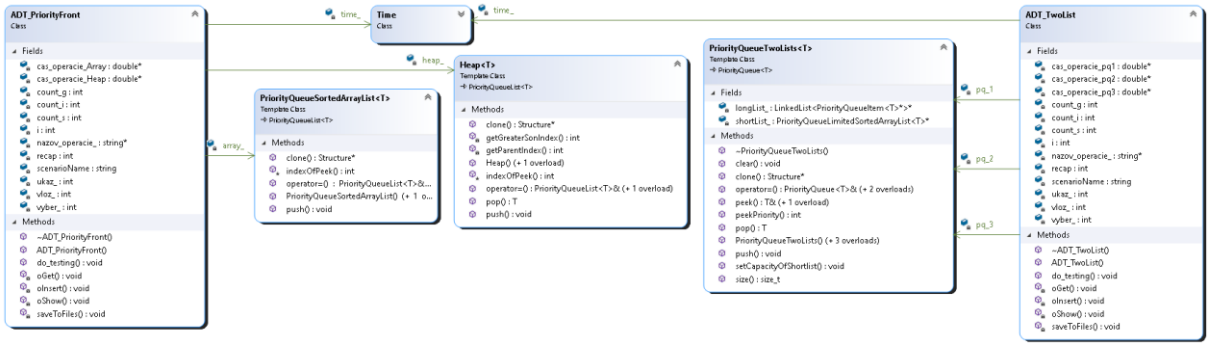
n	Nazov Operacie	Cas LinkedLidst	Cas ArrayList
1.	2. Odstran prvok na indexe	11 [ms]	14 [ms]

Metodika vyhodnotenia

Vyhodnocovanie výsledkov testov bude prebiehať hlavne grafickým porovnaním pomocou programu Excel, prostredníctvom ktorého sa vykreslia grafy pre jednotlivé operácie a vizuálne sa porovná časová náročnosť medzi jednotlivými štruktúrami a určí sa ktorá štruktúra je najvýhodnejšia, a na áke využitie.

ADT prioritný front a Dvojzoznam ako jeho implementácia

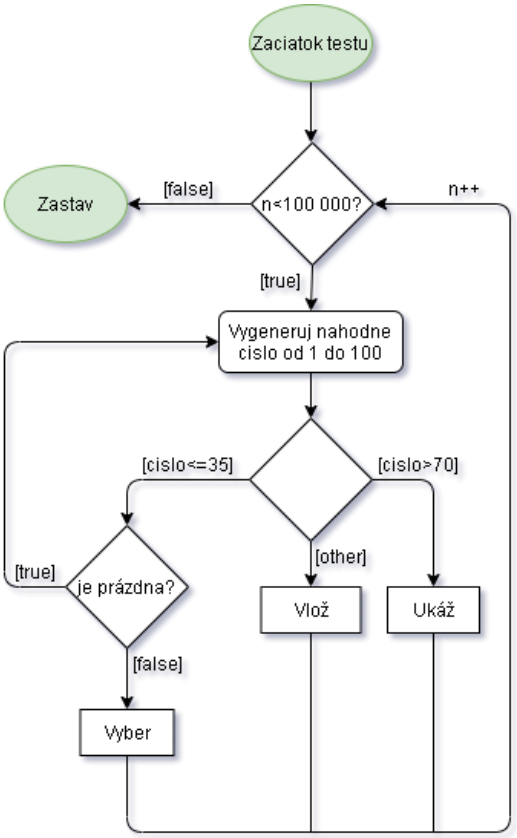
UML diagram návrhu univerzálnych testov



Obe testovacie triedy či už ADT_PriorityFront tak aj ADT_TwoList využívajú rovnaký princíp vykonávania daných scenárov a však jediný rozdiel medzi nimi sú samotné testované štruktúry.

Navrh a priebeh testov

Pred vykonaním každej aktivity sa zadefinuje čas začiatku vykonávania operácie. Vykoná sa príslušná operácia zvoleným algoritmom a ukončí sa vykonávanie. Zadefinuje sa čas konca operácie a za pomoci tohto času sa definuje doba vykonávania príslušnej operácie. Počas celého testovania sa časy trvania operácií uchovávajú a po dokončeného testovania sa zápišu do súborov v danom formáte. Zároveň treba uchovávať aj druh príslušnej operácie pre konkrétny čas, aby sme neskôr vedeli určiť, koľko konkrétnych operácií sa vykonalo a akú mali priemernú dobu trvania.



Formát údajov a súboru

n	Nazov Operacie	Cas ArrayList	Heap
1.	1.Vloz	16 [ms]	14 [ms]

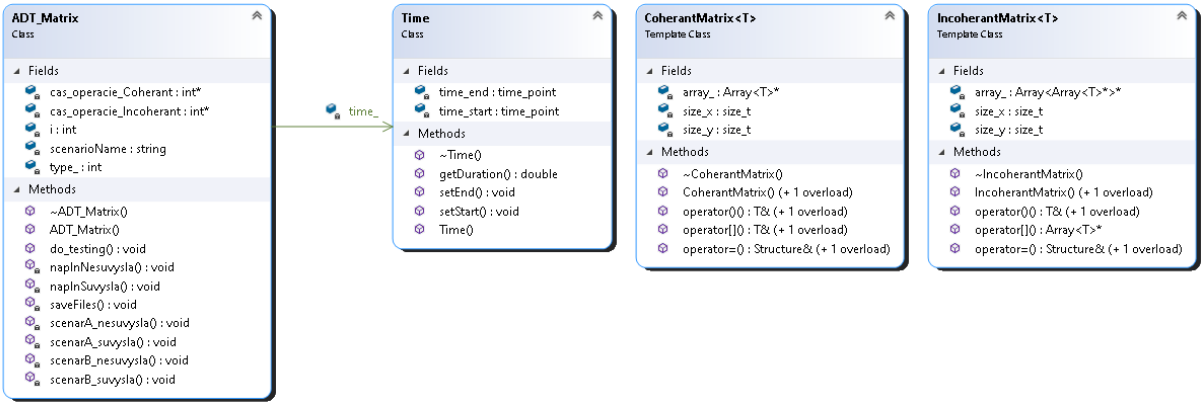
Metodika vyhodnotenia

Vyhodnocovanie výsledkov testov bude prebiehať podobne, ako pri vyhodnocovaní testov pre ADT zoznam, a teda hlavne grafickým porovnaním pomocou programu Excel, prostredníctvom ktorého sa

vykreslia grafy pre jednotlivé operácie a vizuálne sa porovná časová náročnosť medzi jednotlivými štruktúrami.

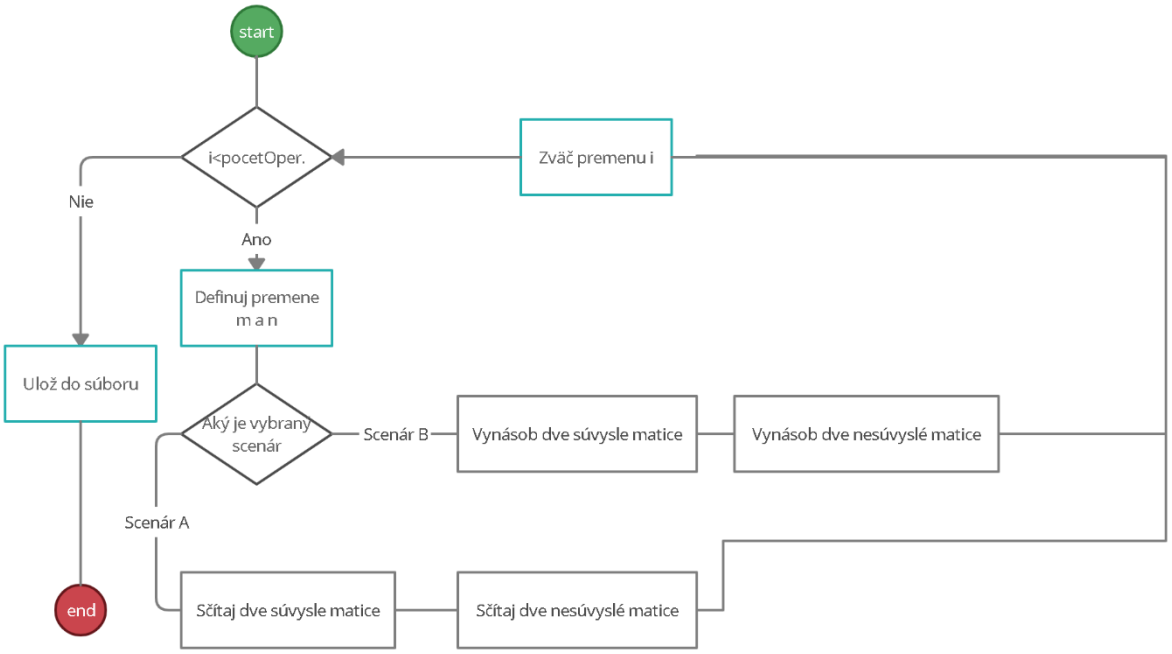
ADT viacrozmerné pole – matica

UML diagram návrhu univerzálnych testov



Počas testovania viacromenrých polí sa jednotlivé matice vytvárajú až v samotných funkciách, ktoré zodpovedajú vykonávanie operácie sčítania a násobenia. Parametre matice sú generované v cykle a sú ukladané do globálnych premenných.

Navrh a priebeh testov



Pri zavolaný funkcie “do_testing()” sa začne cyklus testovania, ktorý vykoná daný počet operácií, ktorý je daný, ako rozdiel maximálne šírky matice a minimálnej šírky matice a keďže chceme porovnávať dva parameter (konkrétne X a Y) treba tento rozdiel vykrátiť dvoma. Po vstupu do cyklu sa definujú parametere matice, šírka X a výška Y na, čo využijeme danú skutočnosť, že počet operácií alebo, teda počet opakovaného cyklu je tvoreným dvojnásobkom šírky, a preto vieme určiť, že počas prvej polovice vykonávania cyklu zväčšujeme iba jeden parameter a druhý je konštantný a neskôr zväčšujeme druhý a prvý je konštantný. Keď už vieme všetky parametre, ktoré sú potrebné, aby sme úspešne vytvorili tri matice rovnakého druhu zistíme, o ktorý test sa jedná či sa jedná o testovanie násobenia dvoch matíc alebo sčitovania a vykonáme danú operáciu pre oba typy matíc.

Formát údajov a súborov

Výsledky testu sa ukladajú do csv súboru vo formáte, ako je nižšie kedy zaznamenávame parameter M a N, ktoré definujú rozmery matice a čas trvania danej operácie (sčitovanie/násobenie).

Pre každý scenár sa vytvorí samostatný súbor v prislúchajúcom priečinku pre daný scenár.

n	M	N	Čas pre maticu vsp	Čas pre maticu vnp
1.	1	K	XY [ms]	XY [ms]
…	…	K	XY [ms]	XY [ms]
M.	M	K	XY [ms]	XY [ms]
M+1.	K	1	XY [ms]	XY [ms]
M+…	K	…	XY [ms]	XY [ms]
M+N.	K	N	XY [ms]	XY [ms]

*konštanta K – náhodná hodnota z interval <1;2000>

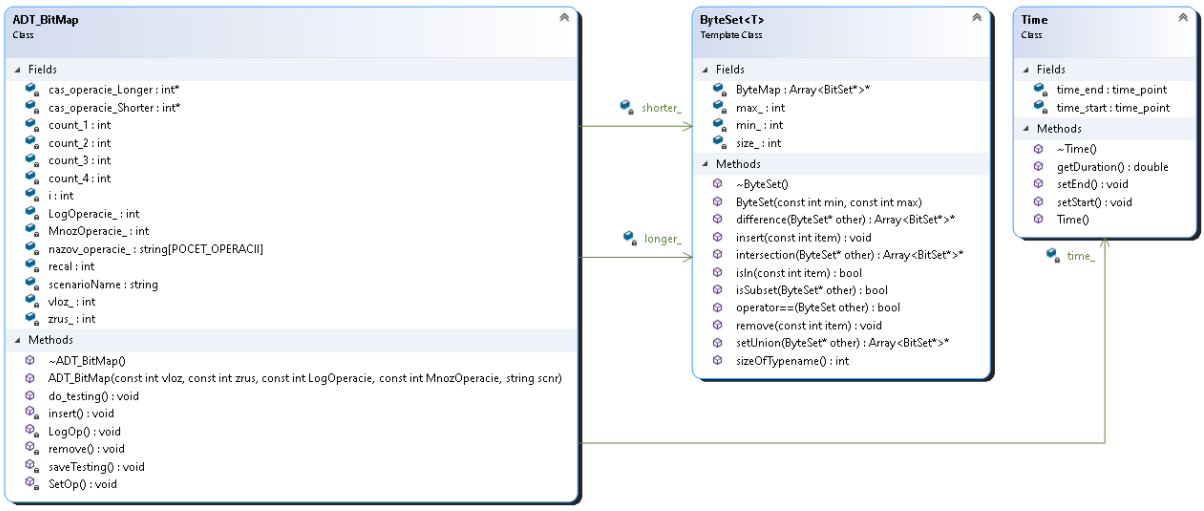
*M, N = 1, 2, …, 2000

Metodika vyhodnotenia

Pre každý scenár využijeme vygenerované súbory a prostredníctvom nich si znázorníme graficky vplyv zmeny parametrov M a N na celkový čas. V grafe využijeme na reprezentovanie Y-ovej osi čas a na reprezentáciu X-ovej osi parameter M a N pričom si ich vykreslíme osobitne, to znamená, že pri každom scenári si budeme musieť vykresliť štyri funkcie.

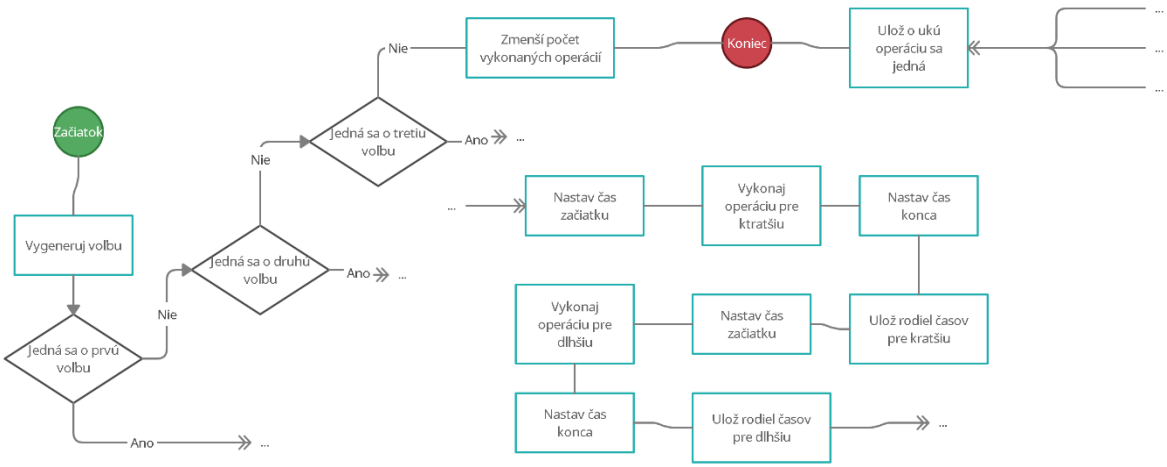
Množina ako bitová mapa

UML diagram návrhu univerzálnych testov



Po zvoleného príslušného scenára si vytvoríme triedu ADT_BitMap s jednotlivými parametrami, pravdepodobnosťami testovaných operácií a zavoláme funkciu do_teting, ktorá zastrešuje celé testovanie. Trieda ByteSet využíva typename, ktoré nám definuje šírku bázovej množiny a naša bázová množina je reprezentovaná ako array bytov, ktorého stĺpce predsatvuju jednotlivé riadky. Po od testovaný si vytvoríme triedu file, do ktorej postupne budeme vkladať jednotlivé výsledky a pomocou tejto triedy ich vyexportujeme do csv súboru.

Navrh a priebeh testov



Keďže aj pri tomto testovaný postupujeme rovnako, ako už pri mnohých iných testoch pozrieme sa bližšie na samotné vnútro skupiny operácií, a teda voľba jednotlivých podskupín a vykonávanie operácií. Pri zavolaný skupiny operácií sa vygeneruje číslo z intervalu, ktorý je definovaný, ako <1;n> pričom n je definovaný ako počet operácií, samotných podskupín. Po generovaný zisťujeme o akú skupinu sa jedná, v prípade ak o žiadnu, zmenšíme počet vykonanných operácií keďže sme žiadnu nevykonali a ukonči funkciu. V opačnom prípade je postup rovnaký, ako môžme vidieť aj v diagrame. Pomocou pomocnej triedy Time si zadefinujeme čas začiatku operácie, vykonáme príslušnú operáciu a znova zavoláme funkciu, ktorá nám zadefinuje čas konca vykonávania operácie. Rozdieľ týchto časov si následne uložíme do poľa, v ktorom budeme uchovávať všetky časy pre príslušnú dátovú štruktúru.

Postup znova zopakujeme pre každú ďalšiu testovanú štruktúru. Po úspešnom ukončený si uložíme do poľa reprezentovaného typom string názov operácie. Keďže v danom kroku sa vykoná rovnaká operácia pre všetky dátové štruktúry stačí nám iba jedno pole na ukladanie názvov. Takýmto spôsobom prebehneme celý cyklus definovaným počtom operácií a zavoláme funkciu, ktorá nám uloží polia časov a pole názvov do súboru, reprezentujúceho výsledky testu.

Formát údajov a súborov

N	Nazov Operacie	Čas Dlhšia báz. množina	Čas Kratšia báz. množina
1.	3. Je rovna.	1[ms]	1[ms]

Metodika vyhodnotenia

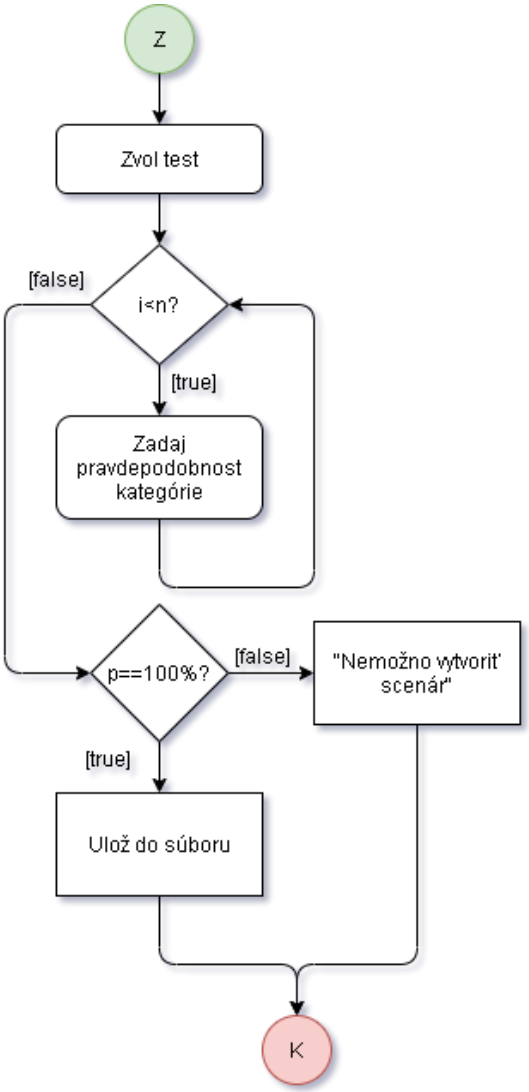
Pri vyhodnocovaní súboru, ktorí dostaneme po úspešného dokončeného testu si dané dáta zoberieme a premiestnime do programu excel. Najskôr si rozložíme jednotlivé operácie, aby sme vedeli o akú operáciu sa jedná a koľko času trvalo vykonanie danej operácie. Rozloženie prebieha tým spôsobom, že pre každú dátovú štruktúru si vytvoríme tri stĺpce zodpovedajúce operáciám Vlož, Vyber a Ukáž a z každej operácie získaný čas zapíšeme do stĺpca, ktorý jej prislúcha. Za pomoci týchto stĺpcov si vypočítame priemer pre každý jeden druh a zároveň aj celkový časový priemer trvania jednej operácie ľubovoľného druhu. Tieto dáta vykreslíme do grafov na základe, ktorých neskôr vyvodíme závery.

Vytváranie scenárov v aplikácií

Popis

Pri vytváraný scenárov si užívateľ zvolí, pre ktoré testy z existujúcich kategórii si želá vytvoriť scenár. Aplikácia začne postupne vyzývať užívateľa, aby zadal hodnotu v percentách pre každú operáciu, ktorá sa v danom teste nachádza. Po zadaní všetkých hodnôt aplikácia skontroluje či boli hodnoty zadané korektne, a teda, či pravdepodobnosť všetkých operáciu dokopy je sto percent. V prípade, že nie, je užívateľ o tejto skutočnosti informovaný.

Diagram



Formát údajov a súborov

Užívateľom vytvorené scenáre sa ukladajú do textových súborov s názvom kategórie, pre ktorú sú určené. V súbore sa na každom riadku nachádza jeden vytvorený scenár a jednotlivé parametere určené uživateľom.

Prezentácia výsledkov

ADT zoznam

Scenár A

Zadanie

Vlož prvý Vlož posledný Vlož na index	Zruš prvý Zruš posledný Zruš na indexe	Sprístupni Nastav	Index prvku
20%	20%	50%	10%

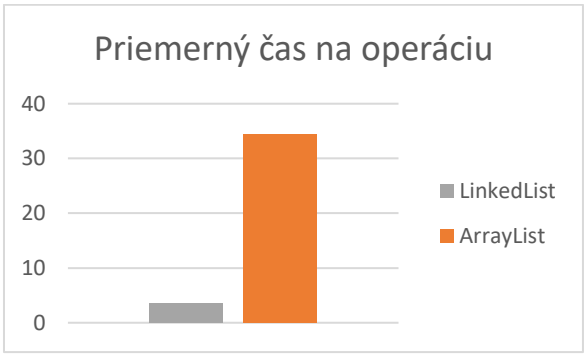
Scenar A - List **podiel operácií v teste*

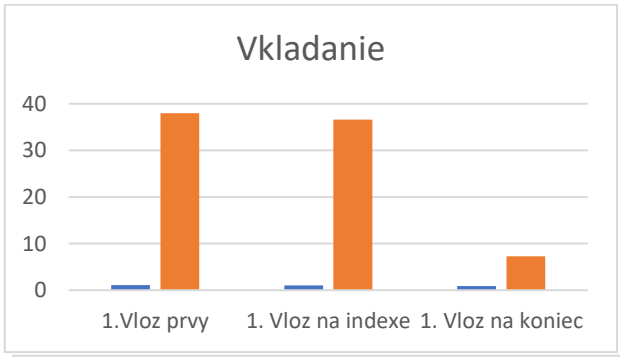
Vysledky

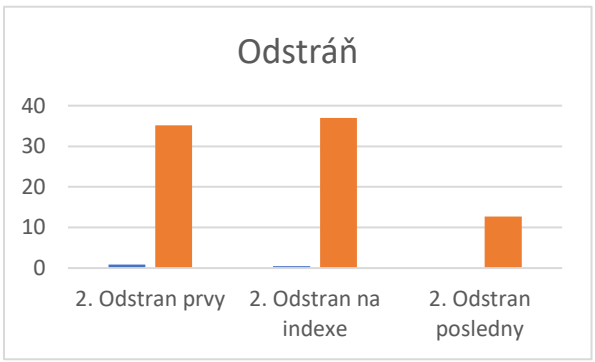
NÁZOV	LINKEDLIST	ARRAYLIST
1.VLOZ PRVY	1.064066	37.97431
1. VLOZ NA INDEXE	1.050666	36.61304
1. VLOZ NA KONIEC	0.85745	7.262748
2. ODSTRAN PRVY	0.437033	35.16961
2. ODSTRAN NA INDEXE	0.396588	36.96784
2. ODSTRAN POSLEDNY	5.654458	12.66052
3. NASTAV	5.233342	12.36244
3. SPRISTUPNI	4.912987	11.42078
4. INDEX PRVKU	3.02607	171.7418
ČASOVÝ PRIEMER NA OPERÁCIU	3.548969	34.47743

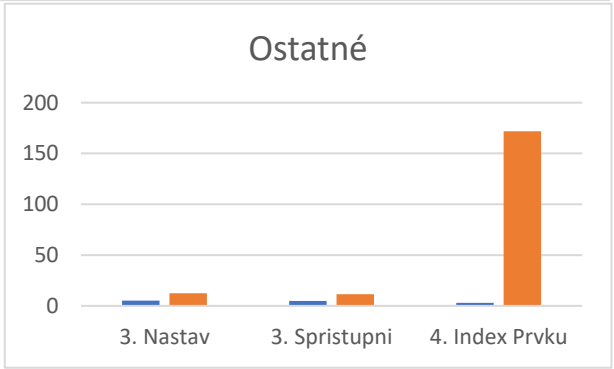
Výsledky A - List **Všetky uvedené hodnoty sú uvedené v microsekundách*

Grafický prehlad výsledkov









Vyhodnotenie

Z testu nám vyplýva, že v prípade, že potrebujeme využiť niektorú zo štruktúr listu a jej použitie bude hlavne na nastavovanie a získavanie jednotlivých prvkov je pre nás najvýhodnejšie využiť dátovú štruktúru LinkedList na rozdiel od ArrayListu.

Scenár B

Zadanie

Vlož prvý Vlož posledný Vlož na index	Zruš prvý Zruš posledný Zruš na indexe	Sprístupni Nastav	Index prvku
35%	35%	20%	10%

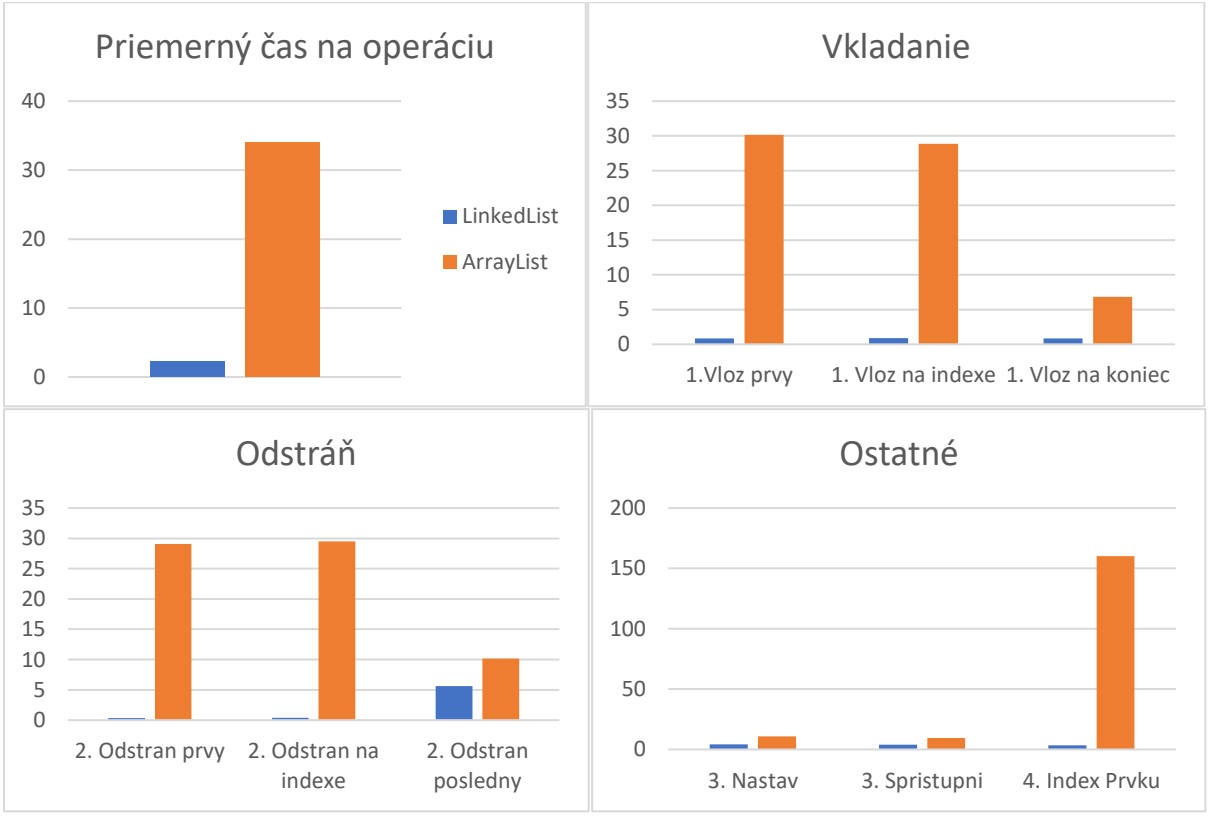
Scenar B - List **podiel operácií v teste*

Vysledky

NÁZOV	LINKEDLIST	ARRAYLIST
1.VLOZ PRVY	0.847899	30.14723
1. VLOZ NA INDEXE	0.898318	28.8593
1. VLOZ NA KONIEC	0.85705	6.814007
2. ODSTRAN PRVY	0.313777	29.09626
2. ODSTRAN NA INDEXE	0.412978	29.5091
2. ODSTRAN POSLEDNY	5.599552	10.15328
3. NASTAV	4.132824	10.93881
3. SPRISTUPNI	3.9	9.482143
4. INDEX PRVKU	3.45625	160.2989
ČASOVÝ PRIEMER NA OPERÁCIU	2.195215	33.98087

Výsledky B - List **Všetky uvedené hodnoty sú uvedené v microsekundách*

Grafický prehlad výsledkov



Vyhodnotenie

Z testu nám vyplýva, že v prípade, že potrebujeme hlavne vkladať a rušiť prvky poľa, ale zároveň chceme raz za čas pri používaní nastavovať, vyberať oplatí sa nám využiť predovšetkým štruktúru LinkedList, ktorá je pre toto použitie výrazne lepšia ako druhú skúmaná štruktúra ArrayList.

Scenár C

Zadanie

Vlož prvý Vlož posledný Vlož na index	Zruš prvý Zruš posledný Zruš na indexe	Sprístupni Nastav	Index prvku
45%	45%	5%	5%

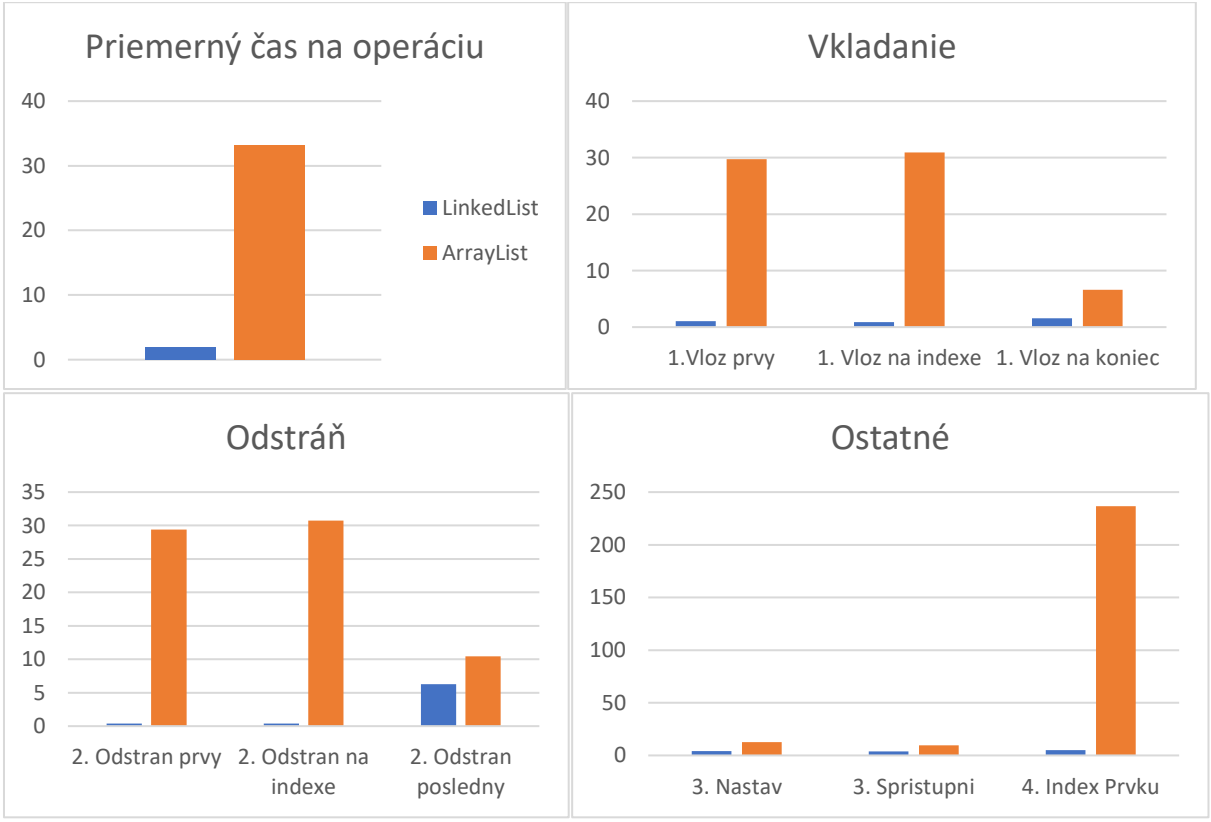
Scenar C - List **podiel operácií v teste*

Vysledky

NÁZOV	LINKEDLIST	ARRAYLIST
1.VLOZ PRVY	1.052158	29.72665
1. VLOZ NA INDEXE	0.886981	30.90342
1. VLOZ NA KONIEC	1.544695	6.629092
2. ODSTRAN PRVY	0.383043	29.3885
2. ODSTRAN NA INDEXE	0.399921	30.72177
2. ODSTRAN POSLEDNY	6.291479	10.44781
3. NASTAV	4.154086	12.65092
3. SPRISTUPNI	3.96	9.5
4. INDEX PRVKU	4.9604	236.6749
ČASOVÝ PRIEMER NA OPERÁCIU	2.015113	33.19323

Výsledky C - List **Všetky uvedené hodnoty sú uvedené v microsekundách*

Grafický prehlad výsledkov



Vyhodnotenie

V porovnaní z prechádzajucími scenármi pri tomto skúmame čisto vkladanie a mazanie prvkov do dátovej štruktúry ArrayList a LinkedList a zas raz nám z testov vyplnule, že aj pre tento typ použitia je oveľa výhodnejšie použiť LinkedList ako ArrayList.

ADT prioritný front

Scenár A

Zadanie

Vlož	Vyber	Ukáž
45%	45%	5%

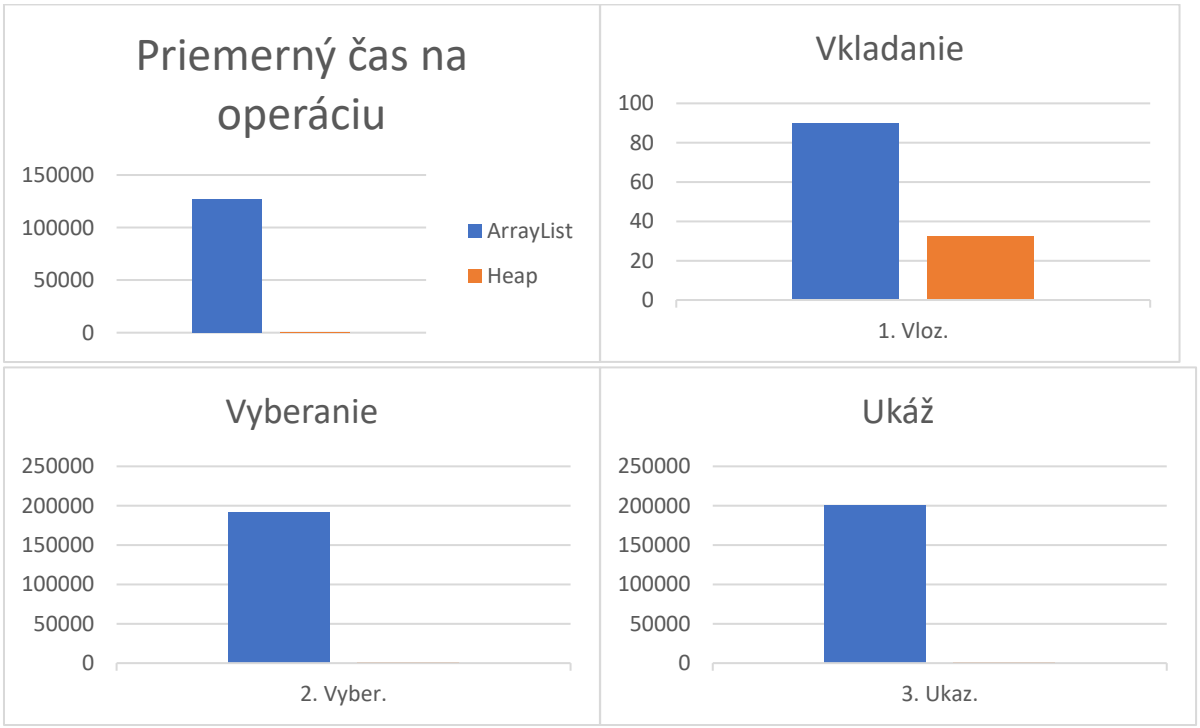
Scenar A -Prioritný front **podiel operácií v teste*

Vysledky

NÁZOV	HEAP	ARRAYLIST
1.VLOZ	89.68849	32.2206
2. VYBER	191167.9	98.80314
3.UKÁŽ	200637.3	13.63145
ČASOVÝ PRIEMER NA OPERÁCIU	127310	45.71648

Výsledky A – Prioritny Front **Všetky uvedené hodnoty sú uvedené v microsekundách*

Grafický prehlad výsledkov



Vyhodnotenie

Pri testovaných testov prioritného frontu skúmame vplyv daných dátových štruktúr na operácie akými sú operácia vkladania do prioritného frontu, vyberania z prioritného frontu a ukazovanie prvkov, ktoré sa v ňom nachádzajú. Zo scenára A nám vyplýva, že bez ohľadu na to ktorú z danej operácie používame je využiť dátovú štruktúru heap.

Scenár B

Zadanie

Vlož	Vyber	Ukáž
50%	30%	20%

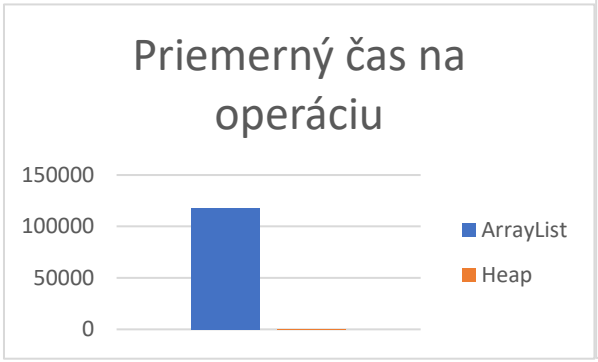
Scenar B -Prioritný front **podiel operácií v teste*

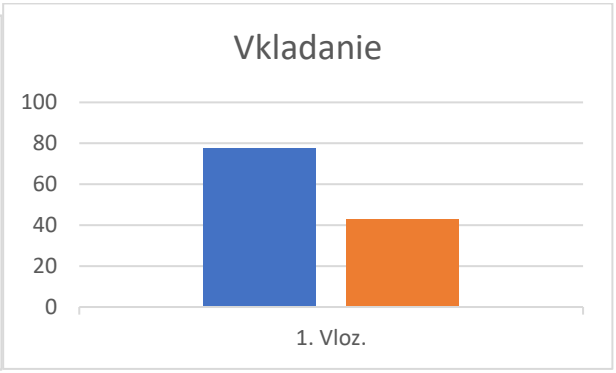
Vysledky

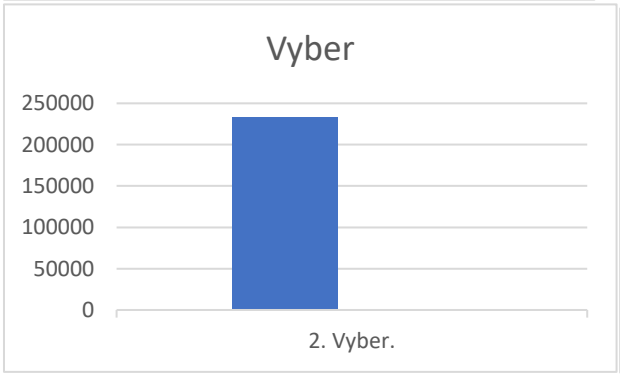
NÁZOV	HEAP	ARRAYLIST
1.VLOZ	77.76299	42.9728
2. VYBER	233229.8	112.6211
3.UKÁŽ	239267	23.92883
ČASOVÝ PRIEMER NA OPERÁCIU	118070.3	51.22069

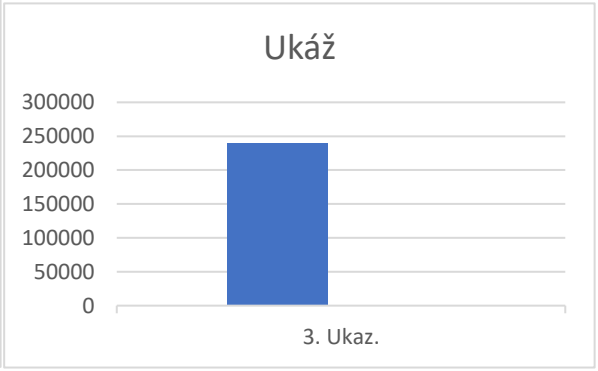
Výsledky B – Prioritny Front **Všetky uvedené hodnoty sú uvedené v microsekundách*

Grafický prehlad výsledkov









Vyhodnotenie

Testovanie scenáru B, ktorý je najvernejšie reprezentuje využitie údajovej štruktúru prioritného frontu v bežných podmienkach nám dokázala, že je na bežné používanie výhodnejšie použiť prioritný front, ktorý je definovaná ako heap.

Scenár C

Zadanie

Vlož	Vyber	Ukáž
70%	25%	5%

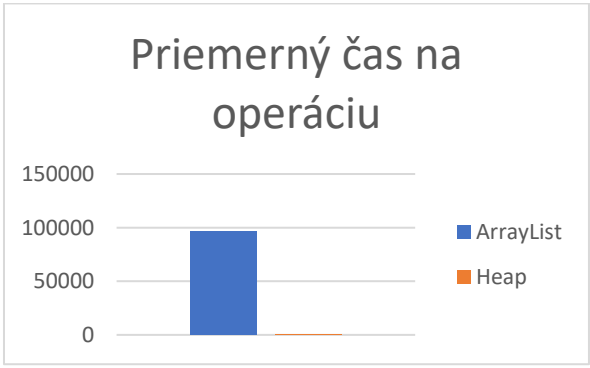
Scenar C -Prioritný front **podiel operácií v teste*

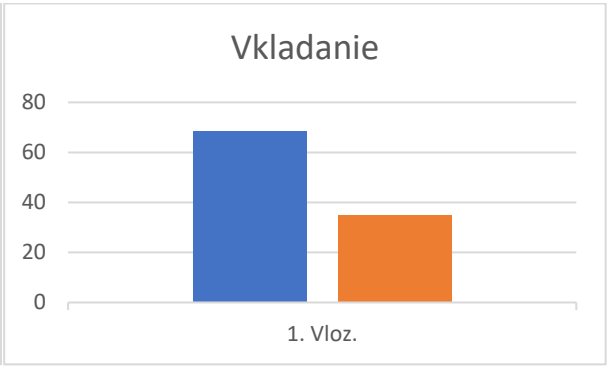
Vysledky

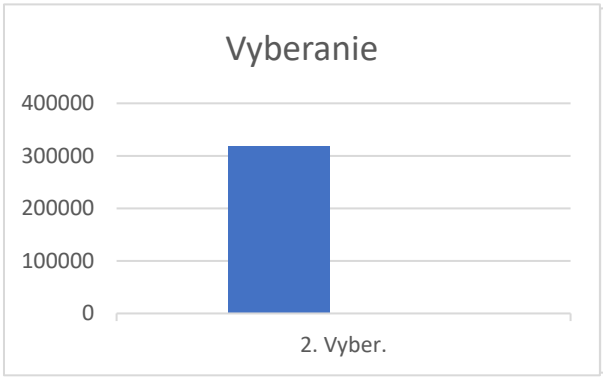
NÁZOV	HEAP	ARRAYLIST
1.VLOZ	68.54369	34.87781
2. VYBER	318656.1	91.4624
3.UKÁŽ	327105.8	19.55457
ČASOVÝ PRIEMER NA OPERÁCIU	97067.2	33.90856

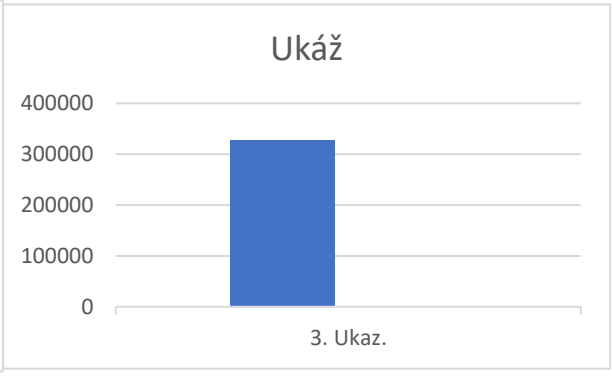
Výsledky B – Prioritny Front **Všetky uvedené hodnoty sú uvedené v microsekundách*

Grafický prehlad výsledkov









Vyhodnotenie

Rovnako, ako pri dvoch predchádzajúcich scenároch taká j pri tomto teste, ktorý bol zameraný hlavne na vkladanie dát do testovaných štruktúr vyšlo, že je najvýhodnejšie v všetkých oblastiach využiť dátovú štruktúru heap.

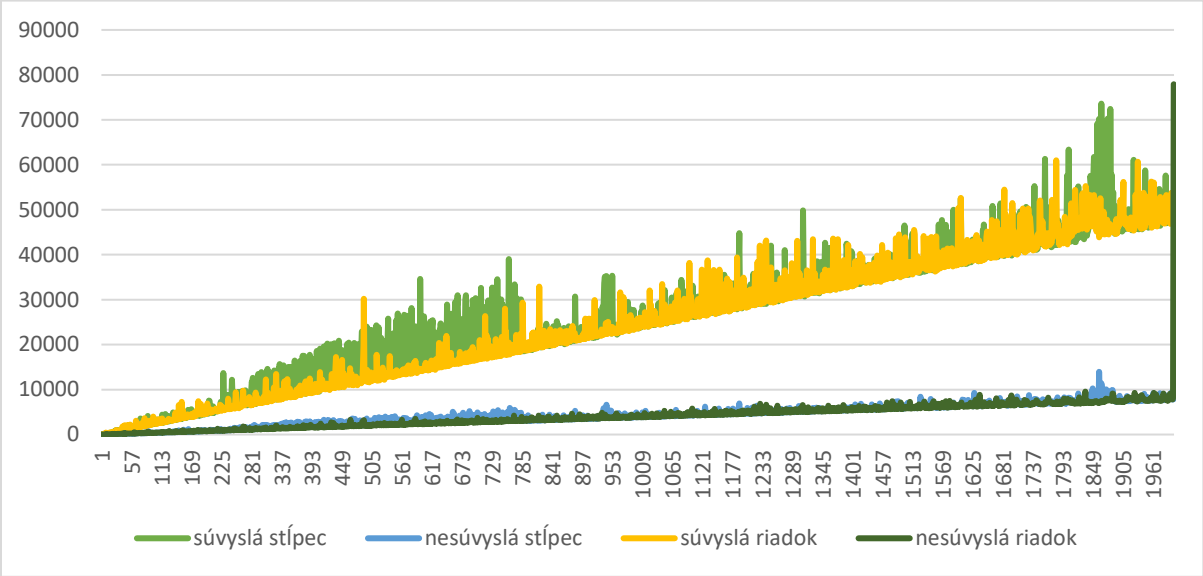
ADT viacrozmerné pole – matica

Scenár A

Zadanie

Vygenerujte dve obdĺžnikové matice rovnakých rozmerov (m x n) a vypočítajte tretiu maticu (rozmerov m x n), ktorá bude ich súčtom.

Grafický prehlad výsledkov



Vyhodnotenie

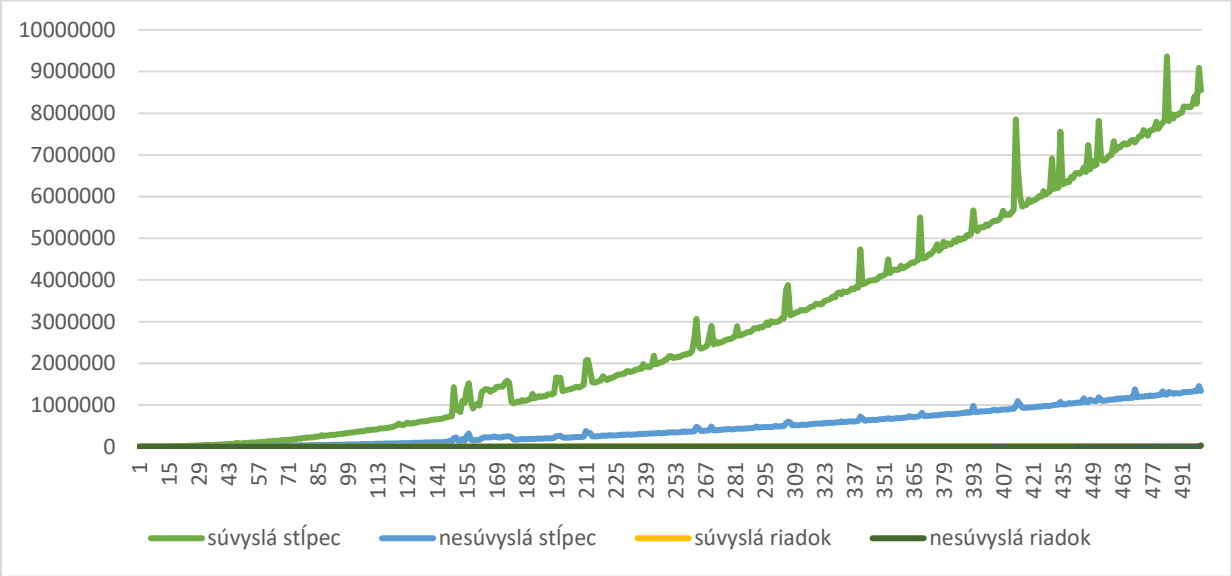
Z testovania nám pomocou grafického znázornenia výsledku vyplynulo, že v prípade sčitovania dvoch matíc a následného uloženia ho do tretej matice rovnakého druhu je výhodnejšie z časového hľadiska využiť maticu definovanú v nesúvislej pamäti, a teda využiť pole polí. Zároveň sa nám ukázalo, že v oboch prípadoch, a teda pri matici definovanej v súvislej, ale i v nesúvislej pamäti je jedno, ktorí prvok M alebo N zväčšujeme.

Scenár B

Zadanie

Vygenerujte dve obdĺžnikové matice rozmerov m x n a n x m a vypočítajte tretiu maticu (rozmerov m x m), ktorá bude ich súčinom.

Grafický prehlad výsledkov



Vyhodnotenie

Pri testovaného súčinu dvoch matíc a následnom presunutí výsledku do treje matice nám vyplynulo obdobne ako v predchádzajúcom teste, že je omnoho výhodnejšie využitie matice, ktorá je daná, ako pole polí, a teda je uložená v nesúvislej pamäti a však sa ukázalo, že v prípade tohto scenára je dôležité či pri daných štruktúrach zväčšujeme index stĺpca alebo riadka, resp. či zväčšujeme šírku matice, alebo jej výšku, a to tak, že je nám výhodnejšie zväčšovať počet prvkov v riadku ako počet stĺpcov.

Obojstranne zreťazený cyklický zoznam

Scenár A

Zadanie

Vlož prvý Vlož posledný Vlož na index	Zruš prvý Zruš posledný Zruš na indexe	Sprístupni Nastav	Index prvku
20%	20%	50%	10%

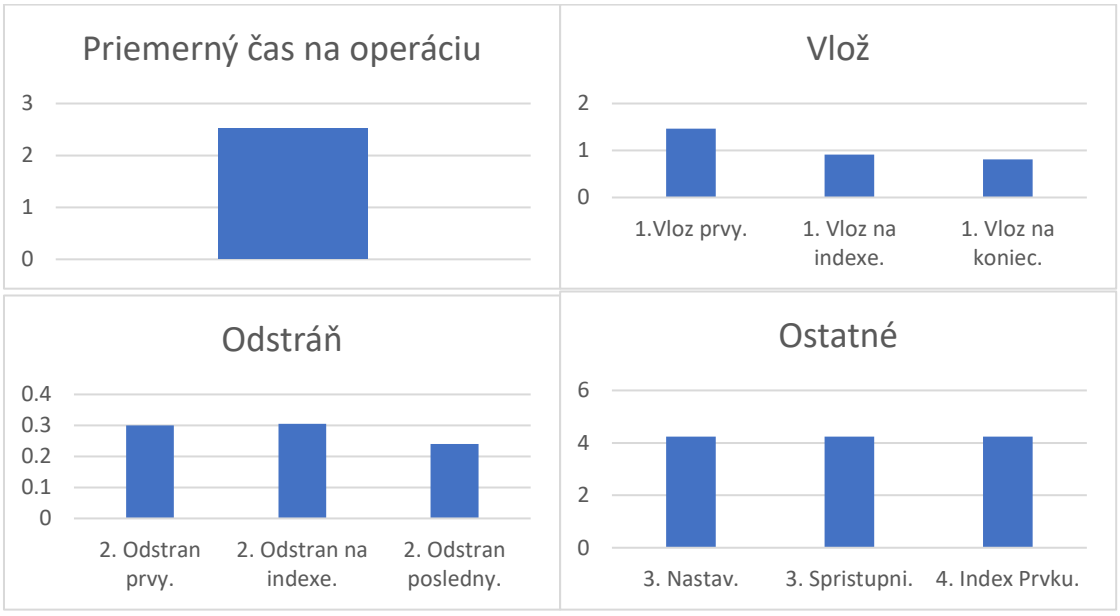
Scenar A - List **podiel operácií v teste*

Vysledky

NÁZOV	LINKEDLIST
1.VLOZ PRVY	1.462765
1. VLOZ NA INDEXE	0.915302
1. VLOZ NA KONIEC	0.809549
2. ODSTRAN PRVY	0.300285
2. ODSTRAN NA INDEXE	0.305516
2. ODSTRAN POSLEDNY	0.240705
3. NASTAV	4.238121
3. SPRISTUPNI	3.953846
4. INDEX PRVKU	1.282777
ČASOVÝ PRIEMER NA OPERÁCIU	2.515626

Výsledky A - List **Všetky uvedené hodnoty sú uvedené v microsekundách*

Grafický prehlad výsledkov



Vyhodnotenie

Aj, keď v grafickom prehľade výsledkov a samotnej tabuľke výsledkov máme zobrazené len údaje pre testovanú dátovú štruktúru DoubleCircularLinkedList realita je taká, že keďže pri tejto štruktúre využívame tie isté testy, ktoré sme používali aj pri úlohe 1, využijeme tieto výsledky a budeme ich porovnávať s týmito na základe čoho nám vyplynulo, že je výhodnejšie využiť obojstranne zretazený cyklický zoznam voči obyčajný zoznam pri všeobecnom použití.

Scenár B

Zadanie

Vlož prvý Vlož posledný Vlož na index	Zruš prvý Zruš posledný Zruš na indexe	Sprístupni Nastav	Index prvku
35%	35%	20%	10%

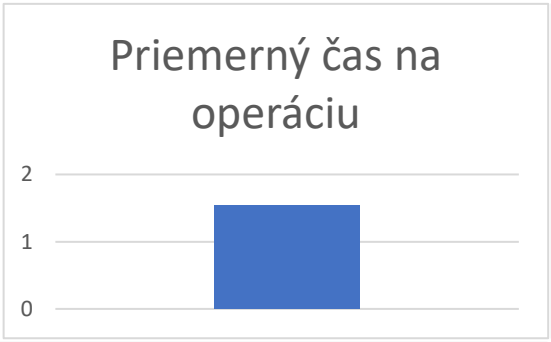
Scenar B - List **podiel operácií v teste*

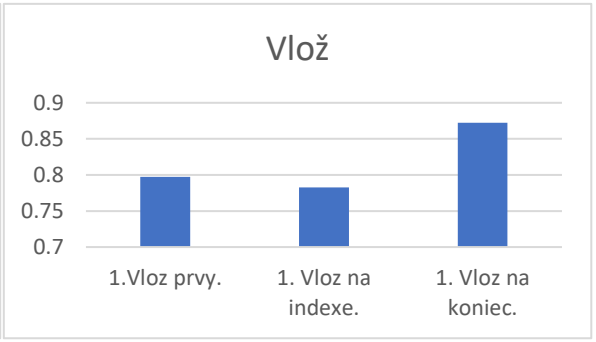
Vysledky

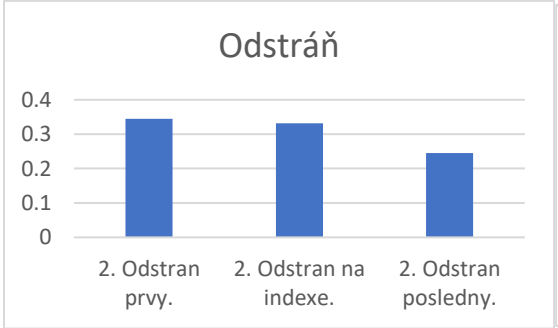
NÁZOV	LINKEDLIST
1.VLOZ PRVY	0.797439
1. VLOZ NA INDEXE	0.782671
1. VLOZ NA KONIEC	0.87238
2. ODSTRAN PRVY	0.344878
2. ODSTRAN NA INDEXE	0.331205
2. ODSTRAN POSLEDNY	0.24533
3. NASTAV	4.464976
3. SPRISTUPNI	3.525926
4. INDEX PRVKU	2.643595
ČASOVÝ PRIEMER NA OPERÁCIU	1.538312

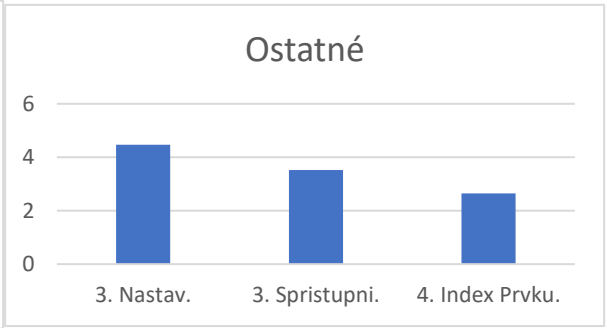
Výsledky B - List **Všetky uvedené hodnoty sú uvedené v microsekundách*

Grafický prehlad výsledkov









Vyhodnotenie

Pri rovnomernom využití operácie štruktúry listu nám vyšlo pri prvej úlohe že je najvýhodnejšie využiť na takéto využitie dátovú štruktúru LinkedList voči druhej testovanej štruktúre ArrayList. Pri pridaný výsledkov z testovania obojstranného zretazeného zoznamu sa nám však situácia mení a zisťujeme že je najvýhodnejšie využť práve tento obojstrane zretazený zoznam.

Scenár C

Zadanie

Vlož prvý Vlož posledný Vlož na index	Zruš prvý Zruš posledný Zruš na indexe	Sprístupni Nastav	Index prvku
45%	45%	5%	5%

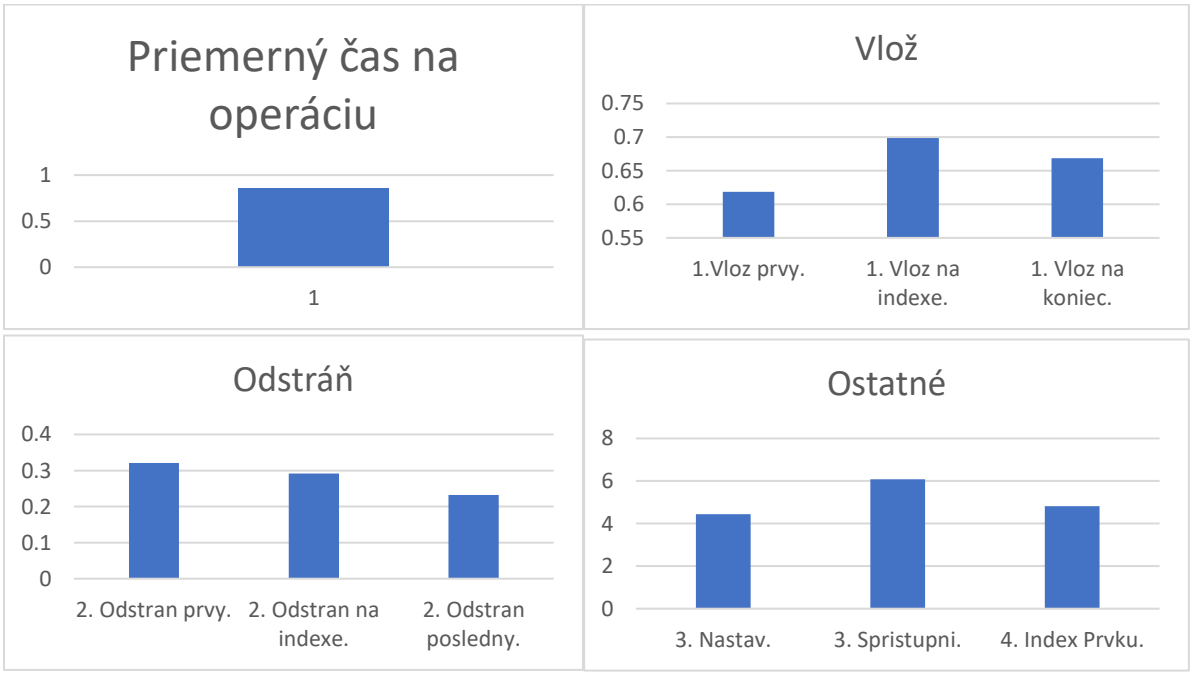
Scenar C - List **podiel operácií v teste*

Vysledky

NÁZOV	LINKEDLIST
1.VLOZ PRVY	0.618337
1. VLOZ NA INDEXE	0.69844
1. VLOZ NA KONIEC	0.668616
2. ODSTRAN PRVY	0.320712
2. ODSTRAN NA INDEXE	0.291365
2. ODSTRAN POSLEDNY	0.232314
3. NASTAV	4.442345
3. SPRISTUPNI	6.075
4. INDEX PRVKU	4.813471
ČASOVÝ PRIEMER NA OPERÁCIU	0.859489

Výsledky C - List **Všetky uvedené hodnoty sú uvedené v microsekundách*

Grafický prehlad výsledkov



Vyhodnotenie

Rovnako, ako pri predchádzajúcich dvoch scenároch tak aj pri scenery, ktorý reprezentuje správanie sa listu, z ktorého iba vyberáme a do ktoré iba vkladáme tak, aj pri tomto použití sa nám najviac oplatí využiť obojstranne zretazený zoznam voči zvyšným dvom štruktúram, aj keď našou druhou nie zrovna najhoršou možnosťou je využitie štruktúry LinkedList.

Dvojzoznam ako implementácia prioritného frontu

Scenár A

Zadanie

Vlož	Vyber	Ukáž
45%	45%	5%

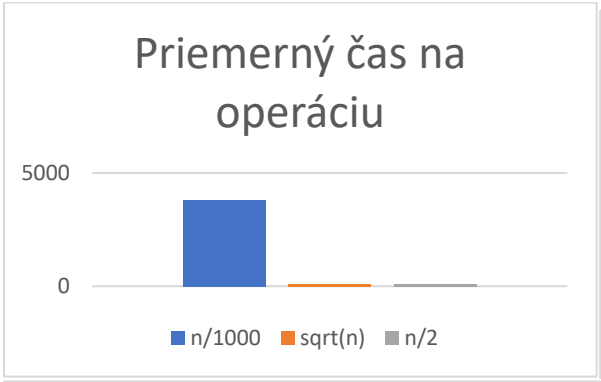
Scenar A -Prioritný front **podiel operácií v teste*

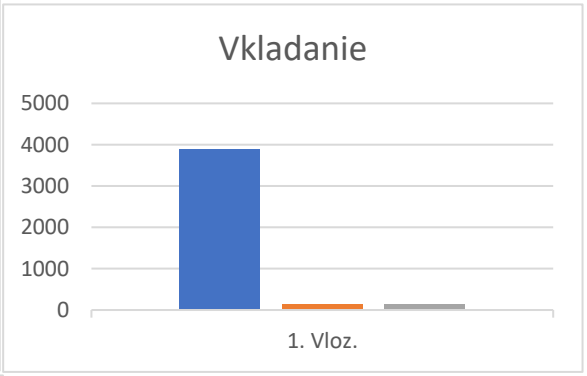
Vysledky

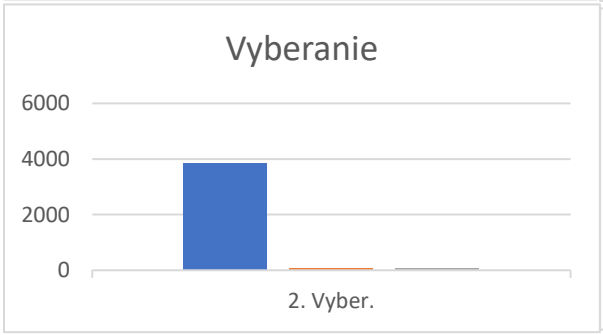
NÁZOV	1/1000	SQRT(N)	N/2
1.VLOZ	3848.032	131.4506	130.3859
2. VYBER	3708.007	87.18463	85.67903
3.UKÁŽ	3848.032	90.91346	84.8605
ČASOVÝ PRIEMER NA OPERÁCIU	3811.172	104.0209	101.0828

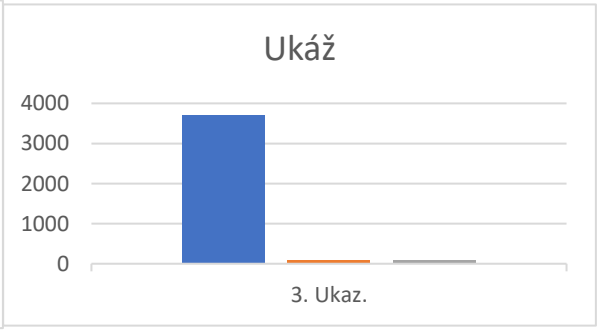
Výsledky A – Prioritny Front **Všetky uvedené hodnoty sú uvedené v microsekundách*

Grafický prehlad výsledkov









Vyhodnotenie

Pri testovaný Dvojzoznamu, ako implementácie prioritného frontu skúmame vplyv dĺžky kratšieho zoznamu na celkové správanie sa dátovej štruktúry. Pri prvom scenery sme testovali správanie sa daných štruktúr pri použití ako náhrada zásobníka, a teda dané prvky do nich buď pridávame alebo z nich odstraňujeme, a teda sa nám ukázalo, že najvýhodnejšie je využiť Dvojzoznam, kde kratší zoznam je definovaný dynamicky. V našom prípade je o kúsok výhodnejšie využiť práve ten, ktorý je definovaný ako polovica z celkového počtu ako využitie kratšieho zoznamu, ktorý je definovaný, ako odmocnina z tohto počtu a však daný rozdiel je len nepatrný.

Scenár B

Zadanie

Vlož	Vyber	Ukáž
50%	30%	20%

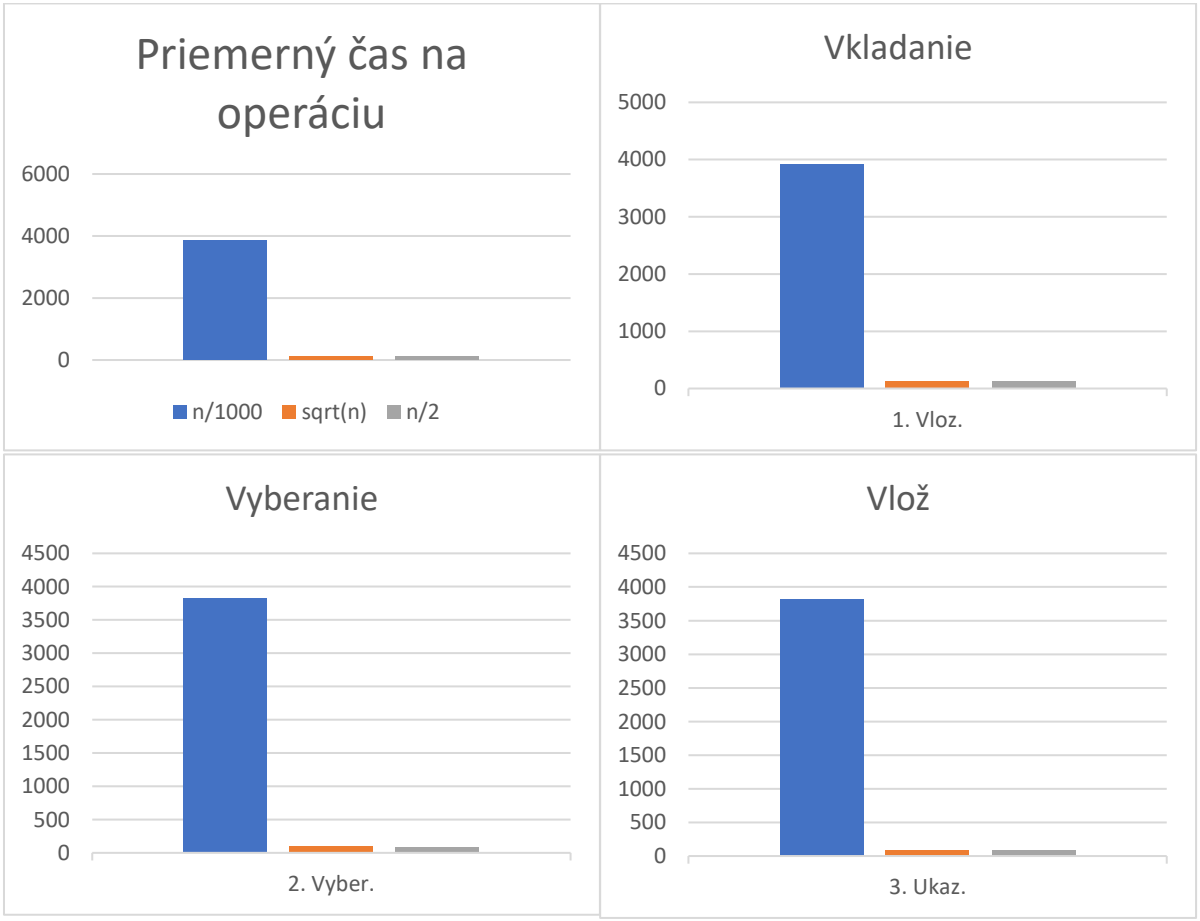
Scenar B -Prioritný front **podiel operácií v teste*

Vysledky

NÁZOV	1/1000	SQRT(N)	N/2
1.VLOZ	3921.565	130.4904	129.0091
2. VYBER	3818.798	95.14993	83.91852
3.UKÁŽ	3815.331	90.32333	87.15436
ČASOVÝ PRIEMER NA OPERÁCIU	3869.298	111.4312	107.4961

Výsledky B – Prioritny Front **Všetky uvedené hodnoty sú uvedené v microsekundách*

Grafický prehlad výsledkov



Vyhodnotenie

Rovnako, ako pri predchádzajúcom prípade sa nám aj v tomto prípade potvrdil rovnaký záver, a to, že je o kúsok výhodnejšie využiť práve ten, ktorý je definovaný ako polovica z celkového počtu ako využitie kratšieho zoznamu, ktorý je definovaný, ako odmocnina z tohto počtu a však daný rozdiel je len nepatrný.

Scenár C

Zadanie

Vlož	Vyber	Ukáž
70%	25%	5%

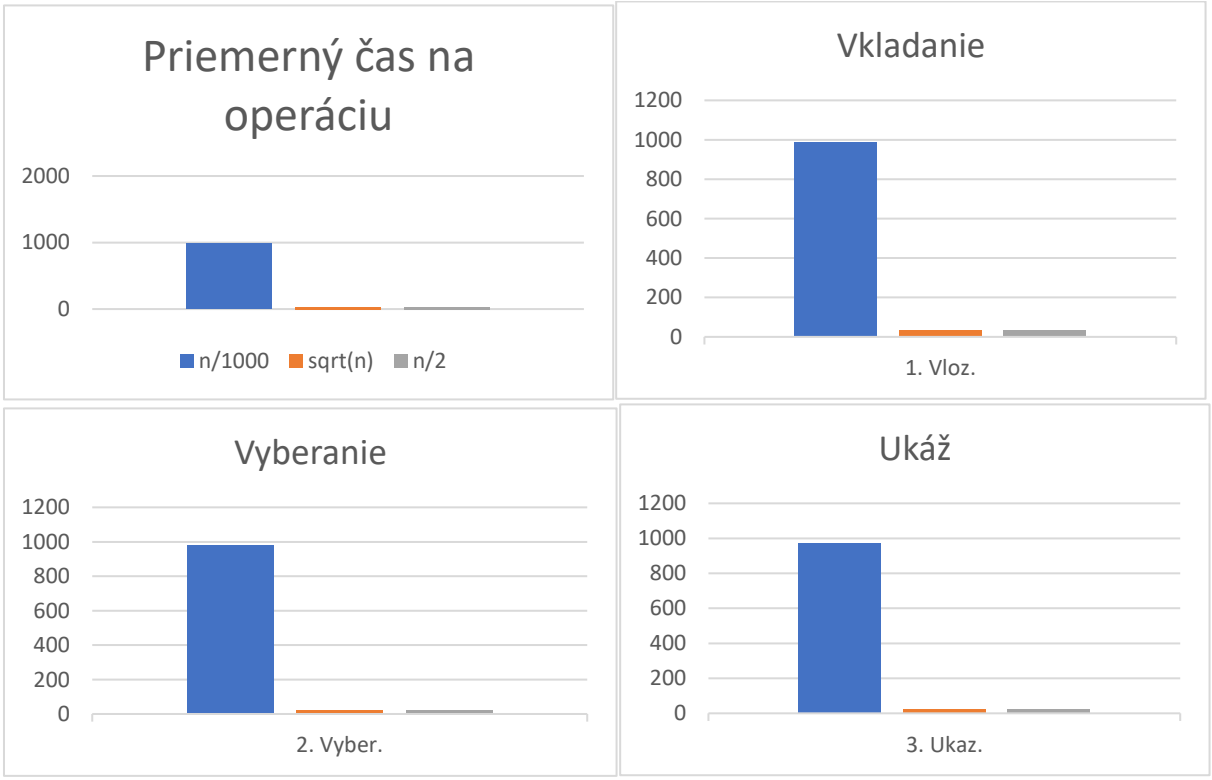
Scenar C -Prioritný front **podiel operácií v teste*

Vysledky

NÁZOV	1/1000	SQRT(N)	N/2
1.VLOZ	985.0059	32.20598	32.03016
2. VYBER	977.468	22.63968	21.8948
3.UKÁŽ	973.7042	22.15593	21.88973
ČASOVÝ PRIEMER NA OPERÁCIU	981.8326	29.24098	29.01434

Výsledky C – Prioritny Front **Všetky uvedené hodnoty sú uvedené v microsekundách*

Grafický prehlad výsledkov



Vyhodnotenie

Testovanie scenára C reprezentuje prípad použitia, kedy prioritný front využívame zväčša na ukladanie dát. V tomto teste sa ukázalo, že konštantne definovaná šírka kratšieho zo zoznamov nieje výhodná a spôsobuje dlhé vykonávanie jednotlivých operácií. V prípade zvyšných dvoch štruktúr, v takomto prípade nieje dôležité aký spôsobom bude kratší zoznam definovaný v prípade, že sa jeho šírka bude dynamicky meniť.

Množina ako bitová mapa

Scenár A

Zadanie

Vlož	Odober	Patrí Je rovná Je podmnožina	Zjednotenie Prienik Rozdiel
20%	20%	50%	10%

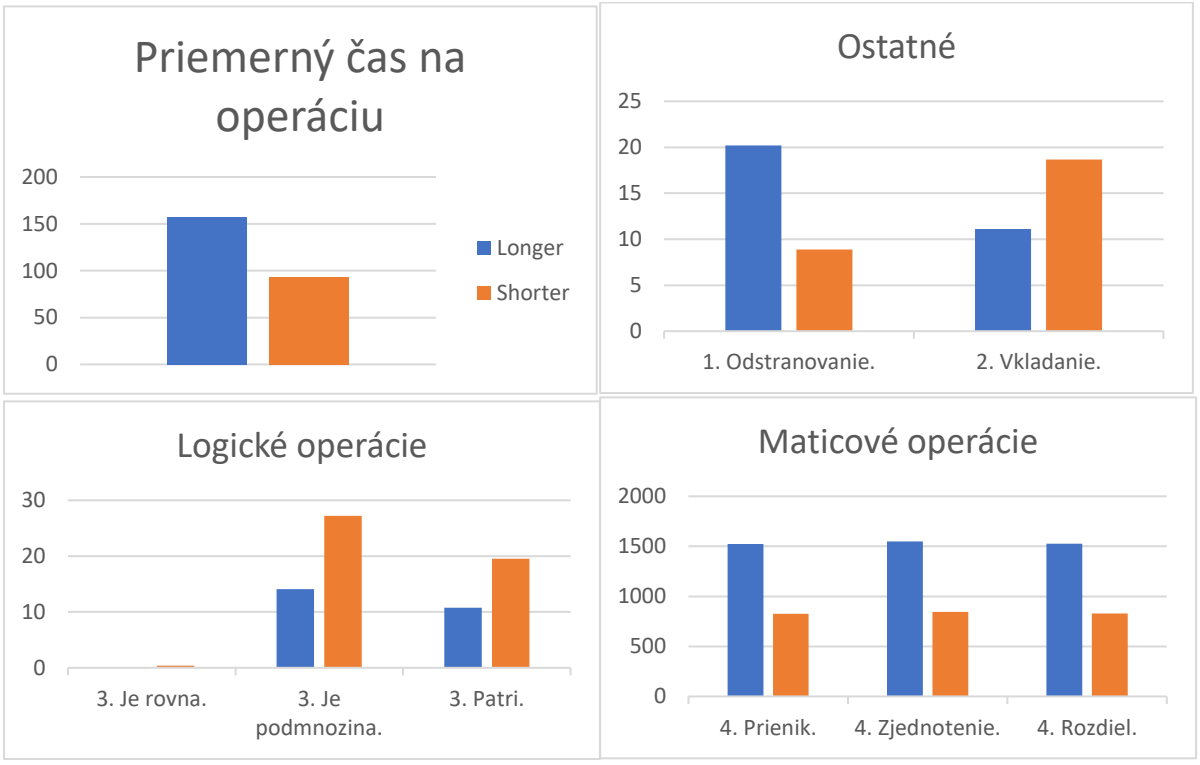
Scenar A - BitMap **podiel operácií v teste*

Vysledky

NÁZOV	SHORTER	LONGER
1.VLOZ	20.1952	8.8851
2. ODOBER	11.10035	18.65464
3. PATRÍ	0.039813	0.372107
3. JE ROVNÁ	14.10382	27.2168
3. JE PODMNOŽINA	10.7761	19.53853
4. ZJEDNOTENIE	1522.268	825.1747
4. PRIENIK	1547.365	844.8732
4. ROZDIEL	1526.115	828.0484
ČASOVÝ PRIEMER NA OPERÁCIU	157.1054	93.15366

Výsledky A - BitMap **Všetky uvedené hodnoty sú uvedené v microsekundách*

Grafický prehlad výsledkov



Vyhodnotenie

Pri rovnomerných využitých operácií, ktoré vie nami vytvorená bitová mapa vykonávať sa nám ukázalo, že je výhodnejšie využiť bitovú mapu, ktorej bázová množina je kratšia ako dlhšia. A však v prípade, že budeme často nastavovať prvky množiny, ktorá je touto bitovou mapou reprezentovaná alebo budeme potrebovať často kontrolovať či daná množina je podmnožinou inej, stojí za zváženie zväčšenie veľkosti bázovej množiny

Scenár B

Zadanie

Vlož	Odober	Patrí Je rovná Je podmnožina	Zjednotenie Prienik Rozdiel
35%	35%	20%	10%

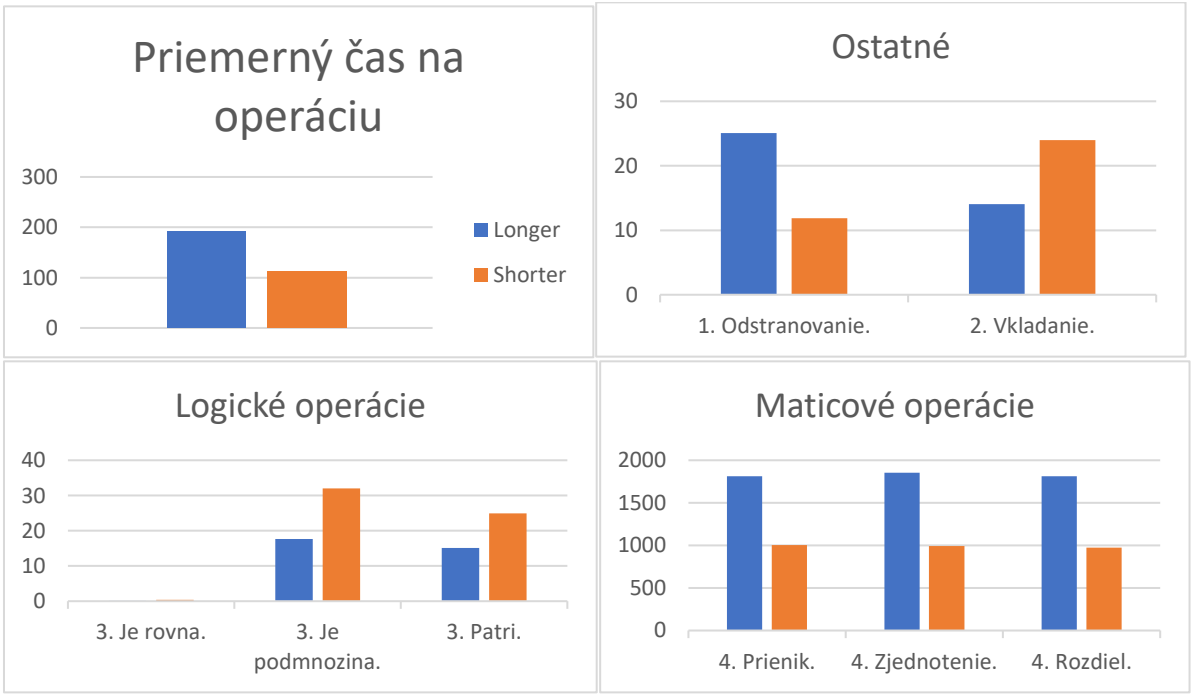
Scenar B - BitMap **podiel operácií v teste*

Vysledky

NÁZOV	SHORTER	LONGER
1.VLOZ	25.0832	11.86197
2. ODOBER	14.03528	23.98958
3. PATRÍ	0.043619	0.386079
3. JE ROVNÁ	17.68418	31.9744
3. JE PODMNOŽINA	15.14962	24.93347
4. ZJEDNOTENIE	1812.208	1003.456
4. PRIENIK	1853.968	993.8654
4. ROZDIEL	1814.117	972.9871
ČASOVÝ PRIEMER NA OPERÁCIU	191.7818	111.7975

Výsledky A - BitMap **Všetky uvedené hodnoty sú uvedené v microsekundách*

Grafický prehlad výsledkov



Vyhodnotenie

Aj z tohoto testovania daným scenárom B nám vyplývajú rovnaké zistenia ako zo scenára A, a to to, že je výhodnejšie využívať Bitovú mapu, ktorá je definovanou kratšou bázovou množinou a však tento záver nie vždycky platí a stojí za zváženie v akom pomere sa budú nami implementované operácie vykonávať.

Záver

Úloha 1 a 4

Z testovania listov sme získali informácie o tom, že spomedzi testovaných odnoží listu je celkovo časovo najvýhodnejšia a zároveň aj pre všetky operácie najrýchlejšia štruktúra obojstranne zreťazeného prioritného listu. Zároveň nám vyplynulo, že s výnimkou operácie odstraňovanie posledného prvku vyšiel Arraylist ako najhoršia dátová štruktúra.

Úloha 2

V rámci testovania u triedeného prioritného frontu a neutriedeného prioritného frontu sa nám ukázalo, že v prípade všetkých troch scenárov a v jednotlivých scenárov aj ich operácií vychádza najvýhodnejšie využiť dátovú štruktúru heap.

Úloha 5

V rámci testovanie dvojitého listu, ako implementácie prioritného frontu prostredníctvom rovnakých testov aké sme využili rovnako aj pri testovaného samostatného prioritného frontu sa ukázalo, že najvýhodnejšia dĺžka kratšieho zoznamu má byť dynamická, a teda sa s počtom prvkov nachádzajúcich sa v prioritnom fronte meniť.

Úloha 3

Pri skúmaní implementácií matíc cez pole polí a jednorozmorné pole sme overovali vplyv druhu implementácie matice a jej šírku s výškou na operácie súčtu a súčinu dvoch matíc.Ukázalo sa, že pri oboch prípadoch je najvýhodnejšie využívať maticu, ktorá je definovaná, ako pole polí. Zároveň pri operácie súčin sa ukázalo, že pri matici definovanej v súvislej pamäti je dôležité dbať aj na výšku matice, a teda na počet riadkov.

Úloha 6

Ak sa bavíme o bitovej mape, jeden z najdôležitejších parametrov je samotná šírka bázovej množiny. V našom prípade sa ukázalo, že s výnimkou logických operácií na množine a zaznamenávanie prvkov, ktoré množina obsahuje sa nám viac oplatí bitová mapa s menšou šírkou bázovej množiny ako so širšou.

Github

Link: https://github.com/Flacore/Testovanie_Struktur/