Fakulta riadenia a informatiky

Informatika

Semestrálna práca S1

# Systém pre geodetov

# 2023

Ing. **Peter Jankovič,** PhD.   
PONDELOK 10, 11 Maroš Gorný, 5ZIB11

2023/2024

Obsah

[Aplikácia na spracovanie údajov zo Sčítania obyvateľov domov a bytov z roku 2021 1](#_Toc105394738)

[Návrh aplikácie z pohľadu použitých údajových štruktúr 5](#_Toc105394739)

[Uloženie územných jednotiek 5](#_Toc105394740)

[Diagram tried a ich popis 6](#_Toc105394741)

[Data 6](#_Toc105394742)

[UzemnaJednotka – Balíček 7](#_Toc105394743)

[Kriterium – Balíček 8](#_Toc105394744)

[Filter – Balíček 11](#_Toc105394745)

[Triedenie – Balíček 13](#_Toc105394746)

[VyberoveKriteria – Balíček 15](#_Toc105394747)

[15](#_Toc105394748)

[Enumy 16](#_Toc105394749)

[Popis výpočtu kritérií 17](#_Toc105394750)

[Zložitosť operácií 19](#_Toc105394751)

[Načítanie dát 19](#_Toc105394752)

[Kritéria 19](#_Toc105394753)

[Filtre 21](#_Toc105394754)

[Triedenie 21](#_Toc105394755)

[Vyhľadávanie najlepších 21](#_Toc105394756)

[Vypísanie údajov 21](#_Toc105394757)

[Používateľská príručka 21](#_Toc105394758)

## Návrh aplikácie z pohľadu použitých údajových štruktúr

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

### Uloženie parciel a nehnutelností

Údajové štuktúry ktoré som použil boli quad stromy, binárný vyhľadaváci strom a taktiež už implementované listy a zásobniky v programovacom jazyku C#. Vzhľadom na to, že sme potrebovali unikátnosť záznamov, vytvorili sme objekty v strome, ktoré niesli dvojicu   
kľúč – dáta. Parcely a nehnuteľnosti sa do stromu pridávali na základe dvoch pozicí a to najviac ľavý dolný roh a najviac horný pravý roh, ktoré presne vymedzili priestor parcely alebo nehnuteľnosti. V strome teda vieme vyhľadávať pomocou daných súradníc.

Quad strom si pamätá maximálnu povolenú hĺbku, vymedzený priestor, kľúče a jedného uzlu, ktorý si pamätá štyri deti - pod uzly, ktoré sú vlastne štyri kvadranty ich otca.   
Tieto uzly majú taktiež vymedzený priestor, ale pamätajú si aj rôzne iné veci, napríklad rez kvadrantov, svoju hĺbku, maximálnu hĺbku v podstrome, svojho otca, prípadne strom v ktorom sa nachádza. Taktiež sú v ňom uložené dáta, typu bod, alebo obdĺžnik.

#### Quad strom

V aplikácií sme urobili tri quad stromy, jeden pre parcely, jeden pre nehnuteľnosti a jeden spoločný, kde sme pridávali aj parcely aj nehnuteľnosti. Vďaka tomuto rozdeleniu, sme mohli efektívnejšie vyhľadávať na základe typu objektu, buď parcely alebo nehnutelnosti.

Spoločný strom nám pomáha pri vyhľadávaní objektu podľa pozície, keď nám nezáleží na jeho type.

Tieto výhody však priniesli aj mierne zhoršenie komplexity pri vymazávaní, pretože vždy musíme objekt vymazat z dvoch stromov. Dopredu však vieme, že operácia vymazať sa bude vykonávať minimálne, preto sme boli ochotní podstúpiť toto mierne zhoršenie.

#### Binárny vyhľadávací strom

Pre potreby obmedzenia duplicitnosť klúčov, si quad strom pamätá všetky kľúče ktoré boli vložené do stromu a to práve v nami implementovanom binárnom vyhľadávacom strome. Na základe tejto informácie, vieme ešte pred vložením, pripadne vymazaním objektu zistiť, či je daný kľúč v strome, alebo nie.

Nakoľko je veľká pravdepodobnosť, že sa nové objekty budú vkladať postupne od najmenšieho kľúču po najväčší, alebo od naväčšieho kľúču po najmenší, môže sa stať, že náš strom zdegeneruje do lineárneho zoznamu. Preto by bolo v budúcnosti vhodné tento strom zmeniť napríklad na Treap, ktorý má oveľa menšiu šancu zdegenerovať na lineárny zoznam.

#### Pole a zásobník

Polia a zásobníky sme si neimplementovali a využili sme už vopred zadefinované implementácie jazyku C#.

Polia sme využívali väčšinou len ako úložný priestor a väčšinou sme v konečnom dôsledku museli prejsť v najhoršom prípade všetky objekty ktoré v ňom boli.

Zásobníky sme využili v prípadoch prehliadky, či už stromov, alebo podstromov.

## Diagram tried a ich popis

Diagram tried si rozdelíme na tri časti z ktorých sa skladá naša aplikácia. Aplikácia je postavená na architektúre MVC (Model – View – Controller) a preto si postupne predstavíme práve tieto komponenty.

### Aplikačná logika (Controller)

Aplikačná logika je kontroler v našej architektúre. Predstavuje teda logiku aplikácie, ktorá spája grafické rozhranie a model. Nachádzajú sa v nej rôzne triedy, napríklad ApplicationLogic, v ktorej sme sa snažili držať hlavnú logiku programu, následne pomocné triedy na generovanie, importovanie alebo exportovanie dát a samotné typy pre nehnuteľnosti a parcely.

V aplikačnej logike máme aj triedy ktoré skúmajú výkon optimalizácie alebo testuju správnosť vytvorených nehnuteľností a parciel. Tieto triedy však pre používateľa nie sú dôležité, preto sme ich do UML diagramu nezahrnuli.

A screenshot of a computer

Description automatically generated

### GUI (View)

GUI je naše grafické rozhranie, takže náš **pohľad**, ktorý je zodpovedný za to, aby používateľový ukázal dáta v korektnom formáte. GUI tiež spracuvavá používateľské vstupy a posiela tieto požiadavky kontroleru.

V grafickom rozhraní máme tri krát Form, tri krát UserControl a následne triedy ktoré nám pomáhajú v navigácii alebo zvýraznení tlačidiel. Form je nové okno a UserControl je len nový obsah v danom okne. Následne tu máme triedu Program, ktorá cele GUI spúšťa.

*MainForm* je trieda, a teda aj okno, v ktorom beží jadro aplikácie, a z ktorého môžeme vykonávať ďalšie akcie, alebo otvárať nové okná, napríklad vyhľadávať nehnuteľnosti, exportovať a importovať dáta, alebo otvorenie okna na generovanie dát.

*GenerateDataForm* je trieda ktorá nám pomáha generovať dáta na základe určeného vymedzeného priestoru a taktiež počtu parciel alebo nehnuteľnosti.

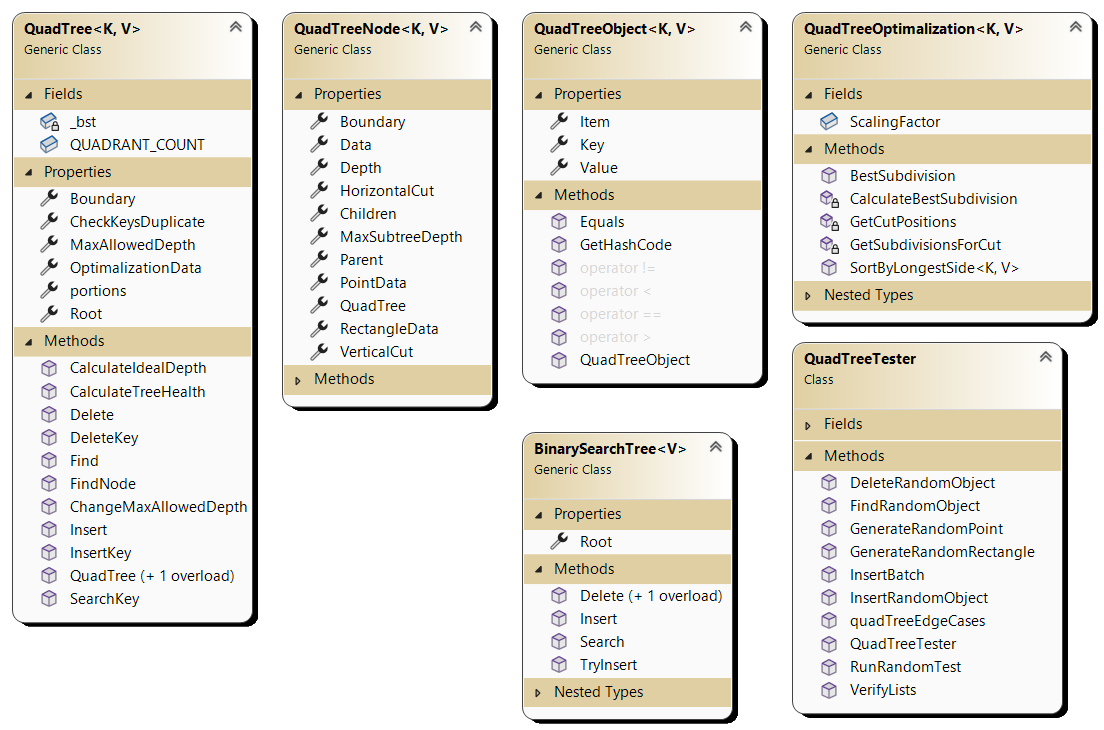
*RealtyEditForm* je trieda určená len na úpravu objektu a teda po otvorení sa nám zobrazia jeho vlastnosti, ktoré máme možnosť upraviť.

A screenshot of a computer program

Description automatically generated

### QuadTree údajová štruktúra (model)

V projekte QuadTree, ktorý je náš model, spracuváva logiku, pravidlá a dáta našej aplikácie. Naimplementovali sme si vlastný QuadTree, jeho Node a Object ktorý do neho vkladáme. Taktiež sme si museli implementovať binárny vyhľadávací strom, vďaka ktorému vieme efektívnejšie ukladať klúče. Okrem iného máme aj triedu na optimalizáciu QuadTree alebo na overenie funkčnosti QuadTree metód.



QuadTree je však založený na priestorových objektoch, bod, alebo obdĺžnik. Vďaka týmto objektom vieme strom rozdeliť na kvadranty. Priestorový objekt si pamätá svoje dve pozície a to pravý dolný roh a pravý horný roh, v prípade bodu, sú tieto pozície rovnaké.

A diagram of a gps point

Description automatically generated

## Zložitosť operácií

### QuadTree

### Aplikačný systém

### Načítanie dát

#### Vloženie obcí

Do utriedenej aj neutriedenej sekvenčnej tabuľky spolu – O(N + log N + V + Z + S)

* N – počet obcí
* V – počet políčok pre vek
* Z – počet políčok pre typ vzdelania
* S – počet políčok pre vek (spočítavanie obyvateľov)

#### Vloženie okresov

Do utriedenej aj neutriedenej sekvenčnej tabuľky spolu– O(N + log N)

* N – počet okresov

#### Vloženie krajov

Do utriedenej aj neutriedenej sekvenčnej tabuľky spolu – O(N + log N)

* N – počet krajov

#### Prepojenie obcí a okresov

O(2\*N + log M )

* N – počet obcí, spočítanie obyvateľov obcí
* M – počet okresov

#### Prepojenie okresov a krajov

O(2\*N + log M )

* N – počet okresov, spočítanie obyvateľov okresov
* M – počet krajov

#### Prepojenie krajov a Slovenska

O(2\*N + M )

* N – počet krajov, spočítanie obyvateľov krajov
* M – počet obci, ktoré si kraj bude pamätať

### Kritéria

#### KUJPrislusnost

O(1)

#### KNazov

O(1)

#### KUJVekPocet

O(R + N)

* R – Vekové rozmedzie (0 až 9 = 10)
* N – počet pod jednotiek

#### KUJVekPodiel

O(R + N + 1)

* R – Vekové rozmedzie
* N – Počet pod jednotiek

#### KUJVekovaSkupinaPocet

O(R + N)

* R – Vekové rozmedzie
  + Predproduktívni = 15
  + Produktívni = 50
  + Poproduktívni = 35
* N – Počet pod jednotiek

#### KUJVekovaSkupinaPodiel

O(R + N + 1)

* R – Vekové rozmedzie
  + Predproduktívni = 15
  + Produktívni = 50
  + Poproduktívni = 35
* N – Počet pod jednotiek

#### KUJNadradena

O(1)

#### KUJTyp

O(1)

#### KUJVzdelaniePocet

O(1)

#### KUJVzdelaniePodiel

O(1 + N + 1)

* N – Počet pod jednotiek

### Filtre

O(N + O(Kriterium))

* N – Počet filtrovaných prvkov

### Triedenie

O(log N)

* N – Počet triedených prvkov

### Vyhľadávanie najlepších

O(N)

* N – Počet prvkov v množine

### Vypísanie údajov

O(N + O(Kriterium) + k)

* N – Počet územných jednotiek ktoré sa majú vypísať
* Kriterium – Kritérium podľa ktorého sa vypíšu údaje
* k – počet kritérií.

## Používateľská príručka

Aplikácia vždy ukáže, aké má používateľ možnosti. Niekedy je treba na konzolu zadať znak, poprípade stačí stlačiť tlačidlo. Ak aplikácia nepovie, že má používateľ niečo stlačiť, čaká na textový vstup. Pri výbere danej obce/okresu/kraja treba napísať správne celý názov územnej jednotky. Pre pomoc sa aplikácia spýta, či chce používateľ vypísať všetky možné územné jednotky s ktorými sa dá pracovať a následne používateľ môže odpísať presný názov územnej jednotky. Pokiaľ sa chce používateľ vrátiť späť, väčšinou platí, že musí napísať charakter „q“ a stlačiť enter.

Pre potvrdenie napísaného textu musí používateľ enter stlačiť dva krát. Ak ho aplikácia vyzýva na stlačenie nejakého tlačidla, ale používateľ chce údaje zadať cez textový vstup, stačí stlačiť ľubovoľné tlačidlo, odporúča sa ale vždy stláčať enter.