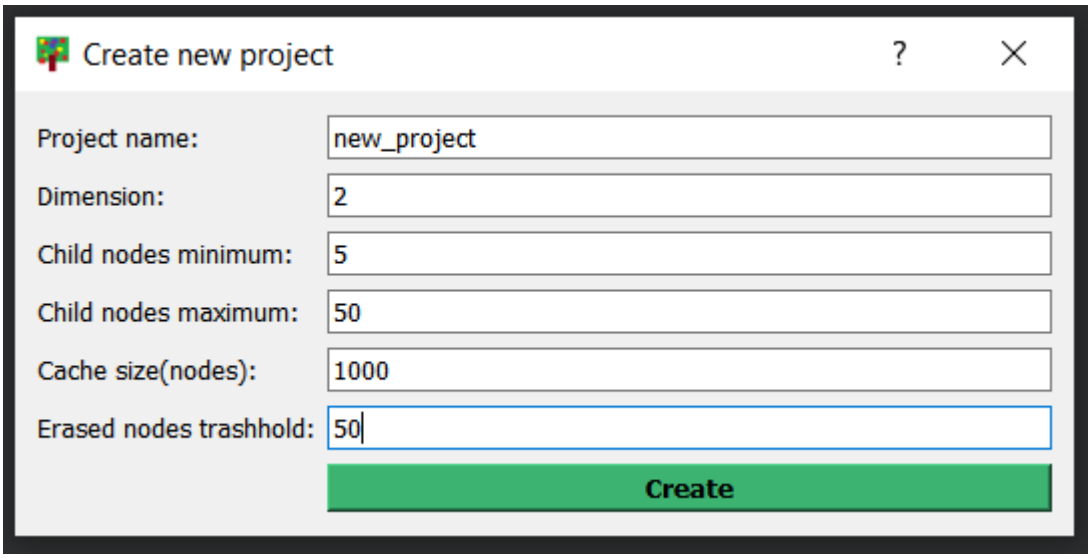
Popis projektu

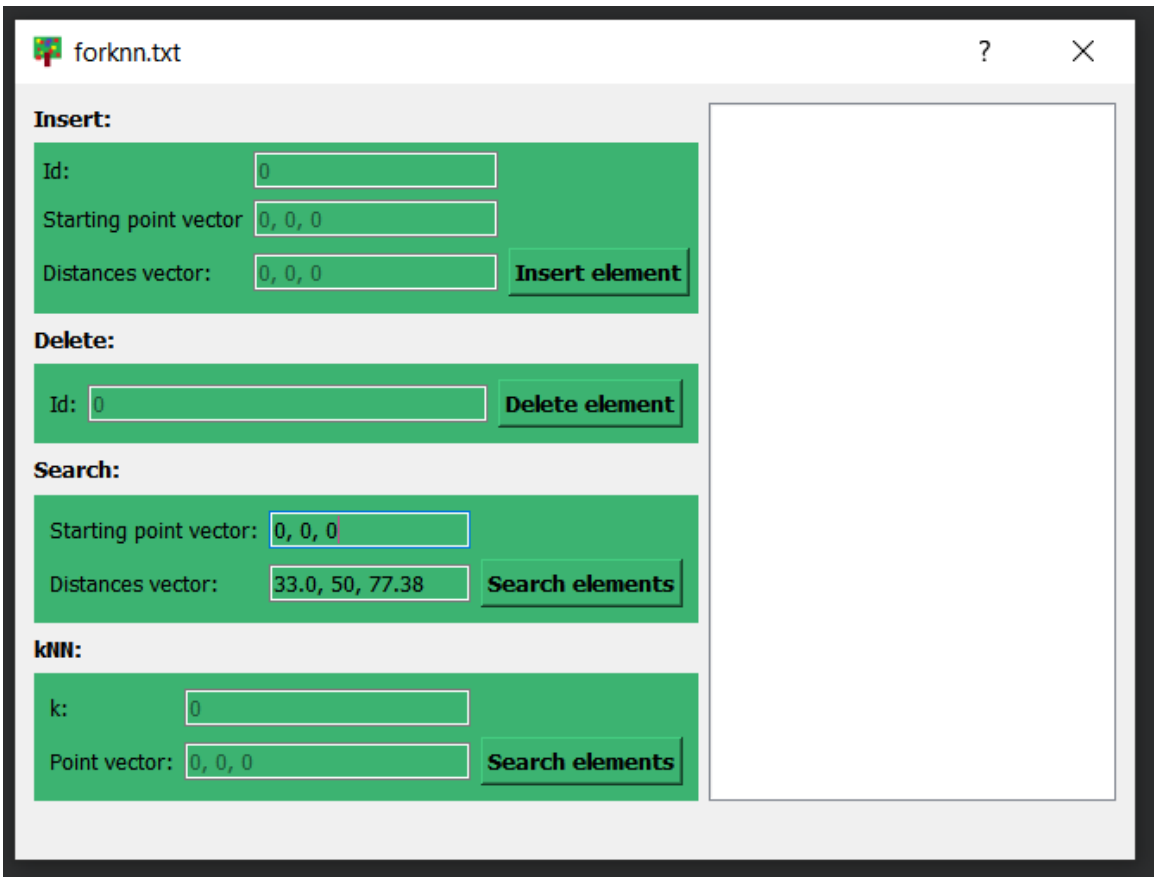
Cílem projektu je implementace datové struktury "R-strom” pro indexování vícerozměrných dat. Celá implementace je založená na původním článku A. Guttmana [1], pojemnování metod odpovídá. Navíc implementace obsahuje k-NN dotaz podle algoritmu z jiného zdroje [2] a jednoduché GUI pro demonstraci.

Vstupy projektu jsou:

- Vytvoření nového projektu: všechny parametry nutné k vytvoření nového projektu.



- Dotazy nad existujícím projektem(Insert, Delete, Search, k-NN): údaje odpovídající určitému dotazu.



Způsob řešení

R strom je implementován podle původní práce Antonina Guttmana [1]. Je zvolen “QuadraticSplit”, který je kompromisem mezi brute-force metodou a zrychlenou lineární metodou. k-NN je implementován na základě metody popsáne v [2]. Jinak je řešení založeno na popisu zadání (https://moodle-vyuka.cvut.cz/mod/page/view.php?id=71690):

R-strom

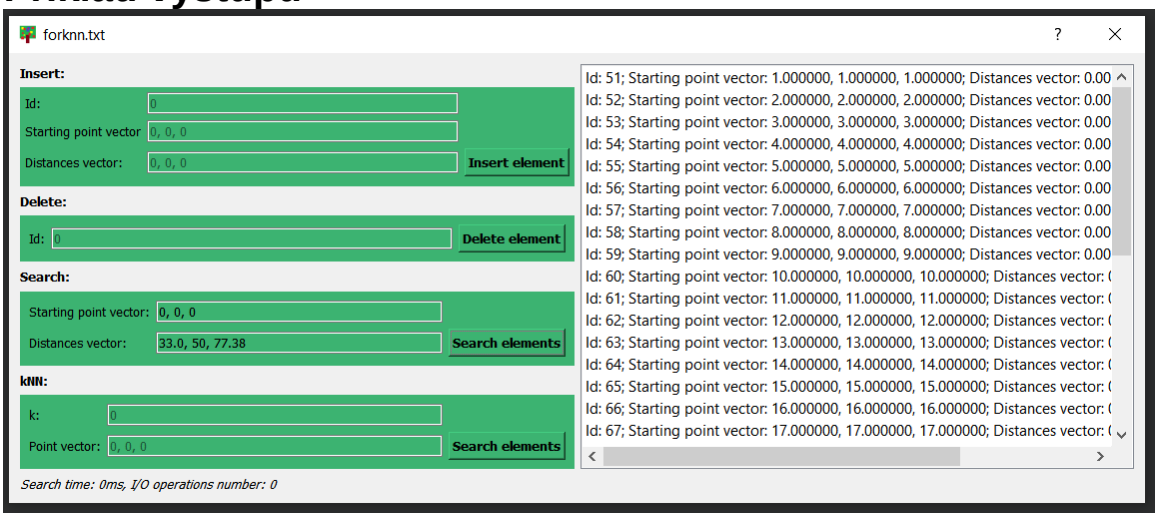
- o R-stromu lze najít spoustu informací na webu (např. https://www2.cs.sfu.ca/CourseCentral/454/jpei/slides/R-Tree.pdf)
- nejprve se vytvoří kořen, do kterého se vkládají jednotlivé záznamy, při přeplnění uzlu dojde k jeho rozdělení na dva a záznamy se rozdělí mezi nimi, oba uzly se pak napojí do nového kořene
- při rozdělení uzlu se používají různé politiky dělení - od náhodného rozdělení záznamů až po testování všech možných rozdělení hrubou silou tak, aby nově vzniklé "obdélníky" byly co nejmenší, což má vliv na rychlost indexování a na rychlost hledání (předpokládá se, že indexujeme jednou za delší dobu a hledáme hodně často)
- strom roste sám vkládáním jednotlivých záznamů a je samovyvažovací
- při vkládání nového záznamu hledáme takový "obdélník", který nebude potřeba zvětšit nebo takový, který bude potřeba zvětšit, co nejméně
- existují různé techniky hledání vhodného listu - single-way (jdeme po jedné větvi), multi-way (procházíme více větví současně)
- při přeplnění listu dojde k jeho rozdělení, při návratu z rekurze dojde i k rozdělení nadřazeného uzlu, pokud je to potřeba, atd.
- při rozdělení kořene strom povyroste o 1 úroveň
- data lze použít náhodně generovaná
- aplikace bude mít možnost definovat počet dimenzí daného n-rozměrného kvádru (obdélník ve 2D, kvádr ve 3D, atd.), vybrat typ dotazu k-NN nebo rozsahový, možnost uložit index do souboru a zase ho načíst
- rozsahový dotaz je definován jako "obdélník"
- u kNN dotazu je na začátku obdélník nekonečně velký, jakmile zaplníme vektor k-nejbližších sousedů, který půjde na výstup, zmenšujeme za běhu jeho rozměry podle podle toho, jak průběžně aktualizujeme položky ve výstupním vektoru
- testovat lze rychlost indexu s ohledem na dimenzi, velikost databáze, apod.
- je potřeba provést porovnání se sekvenčním průchodem databáze (tj. provedení kNN nebo rozsahového dotazu bez R-stromu) a vypsat čas
- pro R-strom pak vypsat čas a ideálně ještě počet uzlů, které bylo potřeba prohledat
- u obou metod je potřeba vypsat výsledek dotazu pro kontrolu
- uzly se serializují na disk (ukládají za sebou do binárního souboru)
- strom bývá hodně široký tj. uzel obsahuje např. 30-50 položek, vhodné je definovat konstantní velikost uzlu např. 2 nebo 4kB a počet záznamů ve vnitřních/listových uzlech z této hodnoty zpětně dopočítat (do listu se vejde více záznamů, protože každá položka zabere méně místa než ve vnitřním uzlu)
- každý uzel má své ID číslováno od 0; v souboru jsou uložena metadata (počet uzlů ve stromu, velikost uzlu, aj.) a za nimi jsou všechny uzly
- pozice (offset) uzlu v souboru se pak vypočte jako "velikost metadat + ID * velikost uzlu"
- strom je optimalizovaný na to, že se nevejde celý do hlavní paměti, můžeme tedy vytvořit vektor (cache), který bude obsahovat určitý počet položek
- pokud se má daný uzel zpracovat, podíváme se nejprve do cache na pozici "ID uzlu modulo kapacita cache", pokud se v ní daný uzel nachází rovnou ho zpracujeme, pokud se v ní uzel nenachází, načteme ho z dané pozice na disku do cache a dále ho zpracujeme
- měřit lze rovněž počet I/O (vstupně/výstupních) operací tj. počet načtení uzlů z disku
- dále lze experimentovat s různým nastavením velikosti uzlů, měřit čas potřebný pro zaindexování databáze, apod.

- mazání záznamů není třeba řešit, v takovém případě je většinou výhodnější záznam jen označit jako smazaný, a pokud je takových záznamů hodně, celou strukturu zahodit a vytvořit novou

Implementace

Celý projekt byl vytvořen v C++ s využitím standardních knihoven, aplikace pro demonstraci byla postavena pomocí knihovny Qt. Pro spuštění na Windows je nutne spustit soubor rtree.exe v adresáři exe.

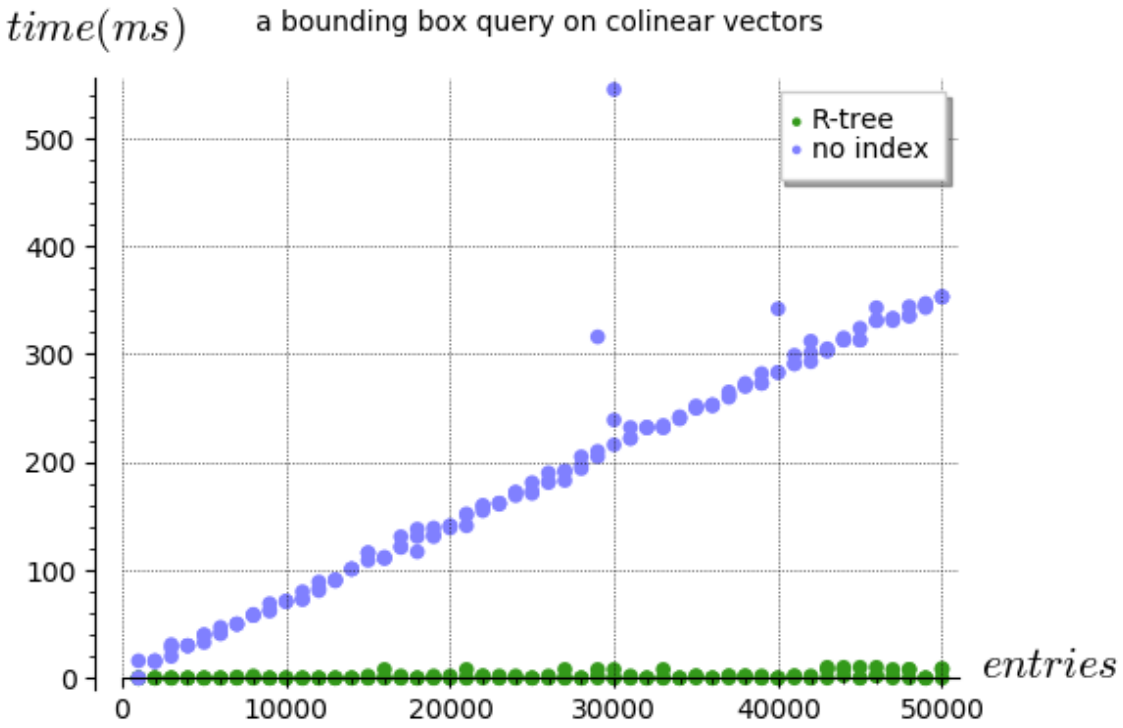
Příklad výstupu

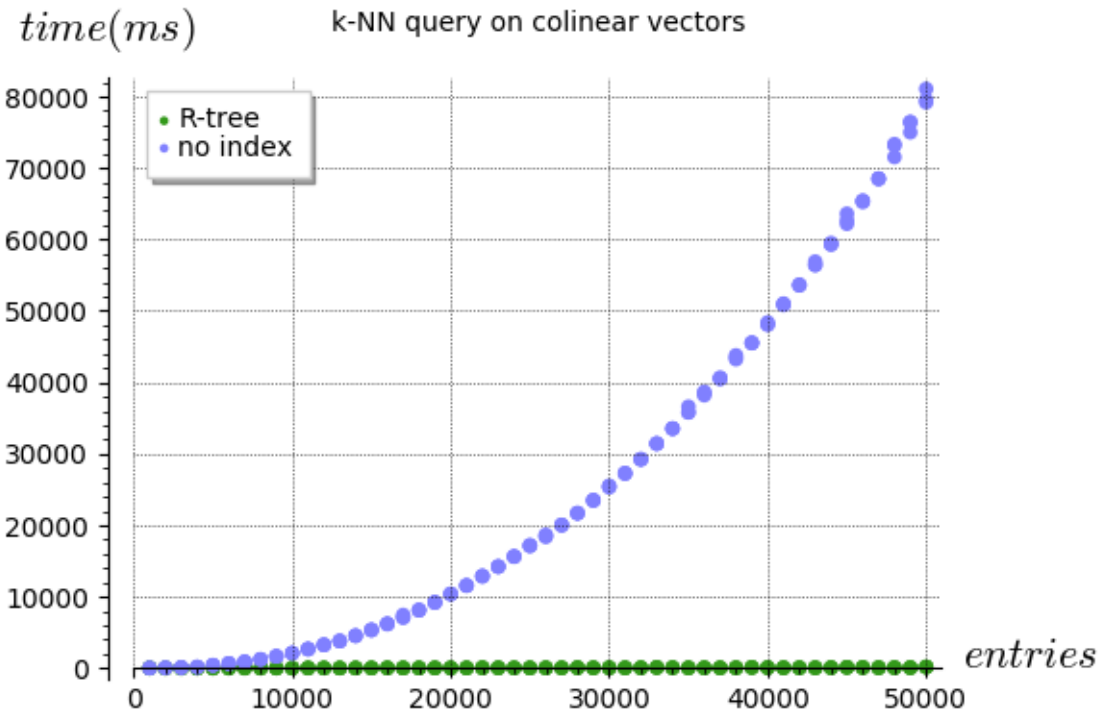


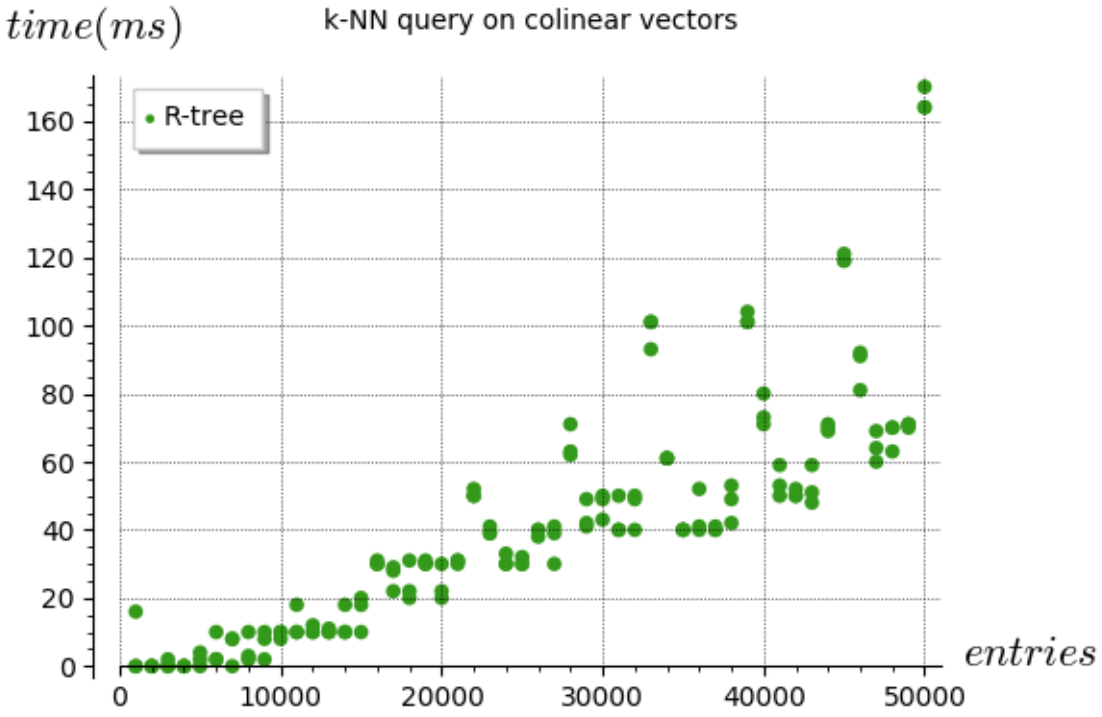
Experimentální sekce

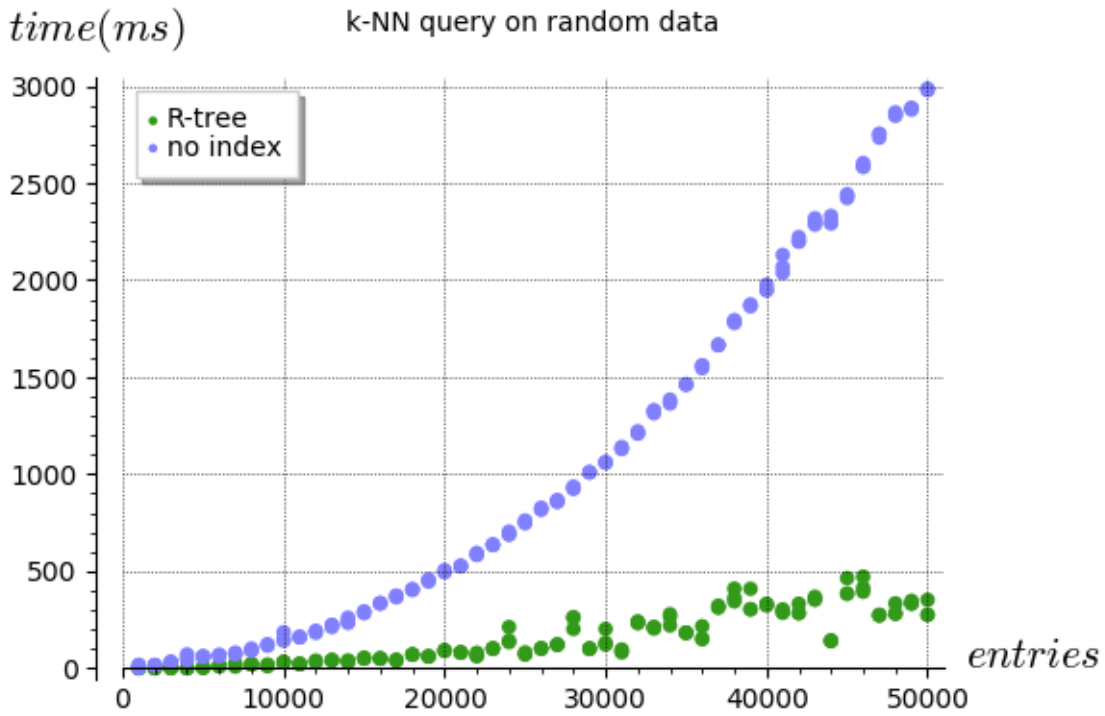
Modul ‘rtreetest’ obsahuje funkce pro testování efektivity R-stromu. Testy využívají přiloženou třídu ‘CNotRTree’, která implemtuje intutivní řešení bez použití indexu. Byly provedeny dva druhy testů: test náhodnými daty a daty, které obsahují nějaké zákonitosti(jako příklad byly zvoleny kolineární vektory). Dimenze dat není podstatná při porovnání algoritmů, proto všechny testy pracují s 3D daty.

Výsledky testů:




time(ms)
k-NN query on colinear vectors
R-tree
no index
80000
70000
60000
50000
40000
30000
20000
10000
0
0
10000
20000
30000
40000
50000
entries


time(ms)
k-NN query on colinear vectors
R-tree
160
140
120
100
80
60
40
20
0
0
10000
20000
30000
40000
50000
entries



Diskuze

Celkově experimenty odpovídají očekáváním zformovaným na základě ználostí o R-stromu. Odhalení nedostatků nebo neočekávaného chování by potřebovalo více testů a experimentů s využitím implementací různých algoritmů a přístupů pro porovnání.

Závěr

Na základě experimentů je R-strom skutečně efektivní datová struktura pro práci s vícedimenzionálními datovými objekty.

Odkazy

[1] R-TREES. A DYNAMIC INDEX STRUCTURE FOR SPATIAL SEARCHING (Antonin Guttman, University of Cahfornia Berkeley)

https://klevas.mif.vu.lt/~algis/DSA/guttman.pdf

[2] Fast k nearest neighbour search for R-tree family(Joseph Kuan, Paul Lewis, University of Southampton)

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.40.3751&rep=rep1&type=pdf