FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

R-strom

Projektová dokumentace k implementaci R-stromu

Toušek, Jan
touseja4@fit.cvut.cz

Prix, Ladislav
prixladi@fit.cvut.cz

May 8, 2019

Vypracováno jako semestrální projekt v předmětu BI-VWM

Katedra softwarového inženýrství
České vysoké učení technické v Praze
Fakulta informačních technologií

Abstract

Tento dokument obsahuje projektovou dokumentaci k semestrálnímu projektu pro předmět BI-VWM. Jde o implementaci R-stromu a webového grafického rozhraní, které umožňuje provádět nad stromem operace vložení, dotazování a ukládání.

1 Popis projektu

Cílem této práce bylo vytvořit vlastní implementaci R-Stromu. R-strom je hyerarchická datová struktura, založena na B+-stromu. Je používán pro indexaci d-dimenzionálních objektů, které jsou reprezentovány jejich minimálním ohraničujícím obdélníkem - Minimum Bounding Rectangle, dále MBR. [1]

Vstupy

Vstupy jsou tvořeny náhodně vygenerovanými daty, nebo daty zadanými uživatelem. Nicméně díky vytvořenému API je možné snadno importovat do stromu jakákoliv data, pokud se jedná o body v d-dimenzionálním prostoru.

Výstupy

Výstupem jsou pak výsledky vyhledávání pomocí různých typů dotazů (Range query, Nearest neighbour query...).

2 Způsob řešení

Objekty, se kterými strom pracuje jsou body v n-dimenzionálním prostoru. Tyto body jsou uloženy v listech stromu. Každý uzel stromu je definován svým MBR, který obsahuje všechny uzly - pokud je uzel vnitřní, nebo body - pokud jde o list.

2.1 Implementované metody

Pro R-Strom jsme naimplementovali následující metody

Insert

Umožňuje vložit nový bod do stromu. Použili jsme algoritmus Insert, popsaný v knize R-trees: theory and applications.[1] Tento algoritmus jsme upravili pro naši potřebu, protože pracujeme s body. Při vkládání nového bodu do stromu záleží na použtém algoritmu dělení uzlů, pokud se některý uzel zaplní. Naimplementovali jsme tři různé algoritmy - lineární, kvadratický a exponenciální. Porovnání těchto algoritmů a jejich výhody a nevýhody jsou blíže popsány v sekci 5.

Contains

Vrátí, jestli strom obsahuje zadaný bod.

RangeQuery

Pro zadaný bod a rozsah vrátí všechny body, které se necházejí v daném rozsahu.

NNQuery

Nearest Neighbour query. Vrátí bod, který se nachází nejblíže zadanému bodu.

KNNQuery

Vrátí k nejbližších bodů zadanému bodu.

3 Implementace

3.1 Architektura

Rozhodli jsme se vytvořit pro R-strom uživatelské rozhraní v podobě webové aplikace. Toto rozhraní komunikuje s API, které na serveru pracuje s daným R-stromem. Obrázek 1

3.2 Zvolené technologie

Backend a API jsme napsali v jazyku C# s využitím frameworku .NET Core. Webová aplikace je vytvořena v JavaScriptu. Pro tyto technologie jsme se rozhodli, protože s nimy máme dobré zkušenosti a protože nám poskytly veškerou potřebnou funkcionalitu. Jako vývojové prostředí jsme použili Visual Studio 2017 Community a Visual Studio Code.

3.3 Použité knihovny

Backend

Newtonsoft.Json – výkonný JSON framework pro .NET[3]

NETStandard.Library – meta knihovna pro .NET core projekty[4]

API

Microsoft.AspNetCore.All – defaultní API pro ASP.NET Core aplikace[5]

Microsoft.NETCore.App – defaultní API pro .NET Core aplikace[6]

Microsoft.Extensions.PlatformAbstractions – abstrakce pro .NET Framework, .NET Core a Mono API[7]

Swashbuckle.AspNetCore – nástroj Swagger pro tvorbu dokumentace API, vytvořených v ASP.NET Core [8]

Frontend

React – JavaScript knihovna pro tvorbu uživatelských rozhraní[9]

Ant Design – knihovna pro grafický design webových aplikací[10]

Konva – knihovna pro kreslení ve 2d[11]

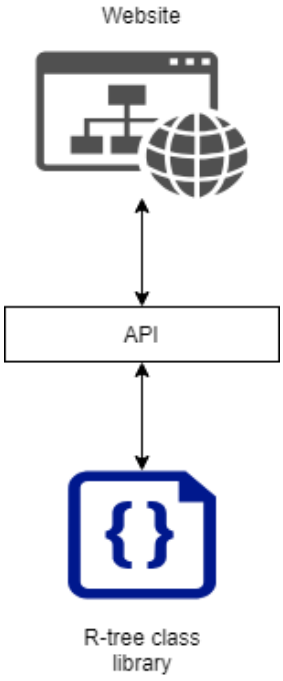

Figure 1: Architektura projektu

3.4 Požadavky pro spuštění

Aby bylo možné projekt spustit, je potřeba mít nainstalované následující:

.NET Core SDK – .NET platforma [12]

Node.js – JavaScript runtime [13]

Node package manager – správce JavaScript balíčků [14]

3.5 Spuštění na platformě Windows

Backend

Pro build backend části je potřeba přesunout se z kořenového adresáře projektu do složky API.

```
cd ./server/Vwm.RTree.Api
```

A poté spustit powershell script BuildAndRun.ps1, který sestaví a spustí projekt API v konfiguraci Release. API je pak defaultně k dispozici na adrese `http://localhost:8080/` 3.5

Frontend

Pro build frontendu se přesunout do složky front a spustit příkazy npm install a npm start. Webová aplikace běží defaultně na adrese `http://localhost:3000/`

Figure 2: Script BuildAndRun.ps1

```
param(
[switch] $noBuild)

if(!$noBuild)
{
   dotnet build --configuration Release
}
dotnet exec .\bin\Release\netcoreapp2.2\Vwm.RTree.Api.dll
```

```
    cd ./front
    npm install
    npm start
```

4 Příklad výstupu

Pro zadávání vstupů a zobrazení výstupů jsme vytvořili webovou aplikaci, která umožňuje používat všechny metody implementované na R-stromu. Níže popíšu záložky, které tato aplikace obsahuje a přidám screenshoty s ukázkami vstupů a výstupů.

4.1 Webová aplikace

Add Point

Umožňuje přidávat body do R-stromu. Obrázek 3.

Dotazy

V rozhraní pro dotazy má uživatel dvě možnosti provedení dotazu.

Perform Query - provede dotaz nad R-stromem, vrátí výsledky a zobrazí dobu trvání dotazu

Perform query with benchmark - provede dotaz nad R-stromem i lineárně a zobrazí oba výsledky a čas trvání obou dotazů pro možné porovnání

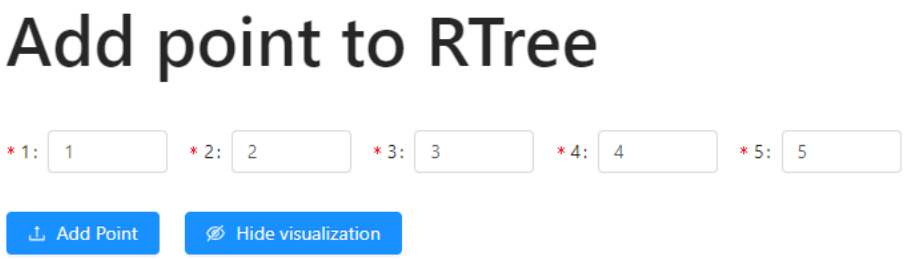

Figure 3: Záložka Add Point

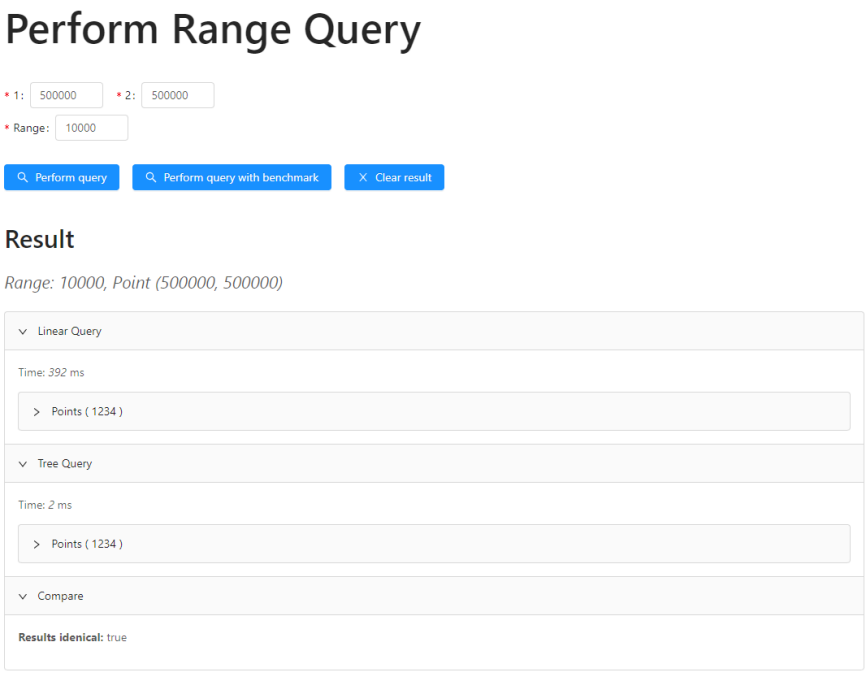

Figure 4: Záložka Range Query

Range Query

Umožňuje zadat bod a rozsah, ve kterém se mají nacházet výsledky. Vyhledané výsledky vrátí a zobrazí. Obrázek 4.

NN Query

Umožňuje zadat bod a vyhledat bod, který je k němu nejblíže. Obrázek 5.

KNN Query

Umožňuje zadat bod a vyhledat k bodů, které jsou k němu nejblíže. Obrázek 6.

Contains

Prohledá R-strom a zjistí, jestli obsahuje zadaný bod.

Data

Na záložce data je možné zobrazit strukturu stromu, včetně bodů, které strom obsahuje. Obrázek 7. Dále zde najdeme tlačítko Replace with fresh, které zobrazí formulář pro vytvoření nového stromu. V tomto formuláři můžeme specifikovat parametry stromu, jako je počet dimenzí, maximální počet uzlů v jednom uzlu, maximální počet bodů v listovém uzlu. Dále můžeme náhodně vygenerovat zadaný počet uzlů. Obrázek 8.

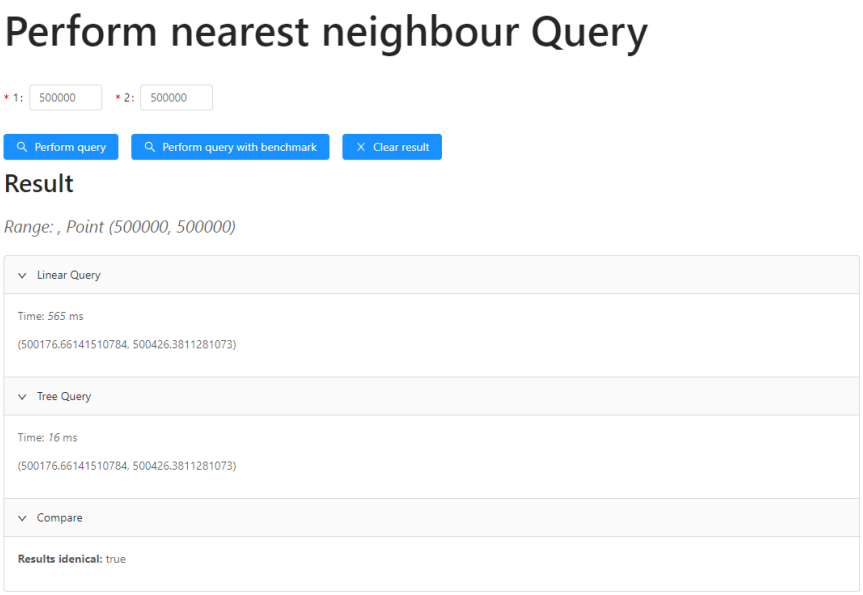

Figure 5: Záložka NN Query

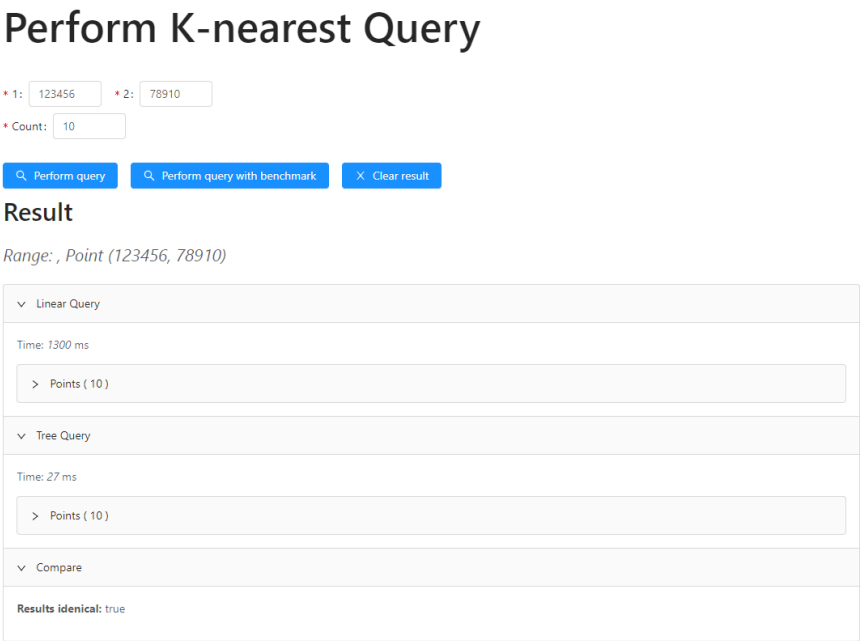

Figure 6: Záložka KNN Query

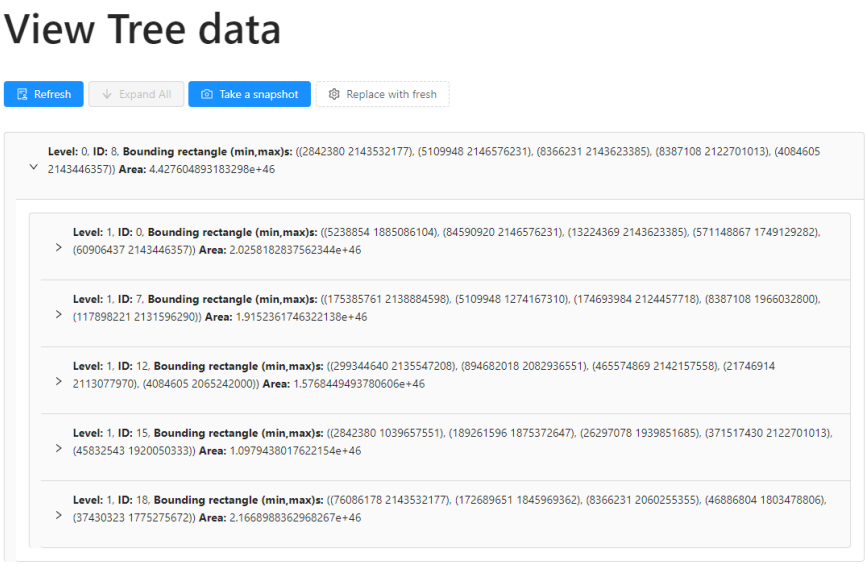

Figure 7: Záložka Data

Tlačítkem Take a snapshot můžeme uložit aktuální stav stromu na disk a s uloženým snapshotem poté dále pracovat na záložce Snapshots.

Visualize

Na záložce visualize je kanvas, který graficky zobrazuje body stromu a MBR uzlů, pokud je strom dimenze 2. Tato záložka zobrazuje, jak jsou body uloženy v uzlech a jak jsou postupně vytářeny další uzly a obdélníky. Na této vizualizaci lze pozorovat rozdíly mezi jednotlivými použitými metodami pro rozdělení plných uzlů. V příkladu na obrázku je použitý Quadratic split. Obrázek 9.

Snapshots

Záložka Snapshots tvoří rozhraní pro práci s uloženými Snapshoty stromů. Snapshot stromu je možné načíst, smazat, nebo stáhnout v JSON formátu.10

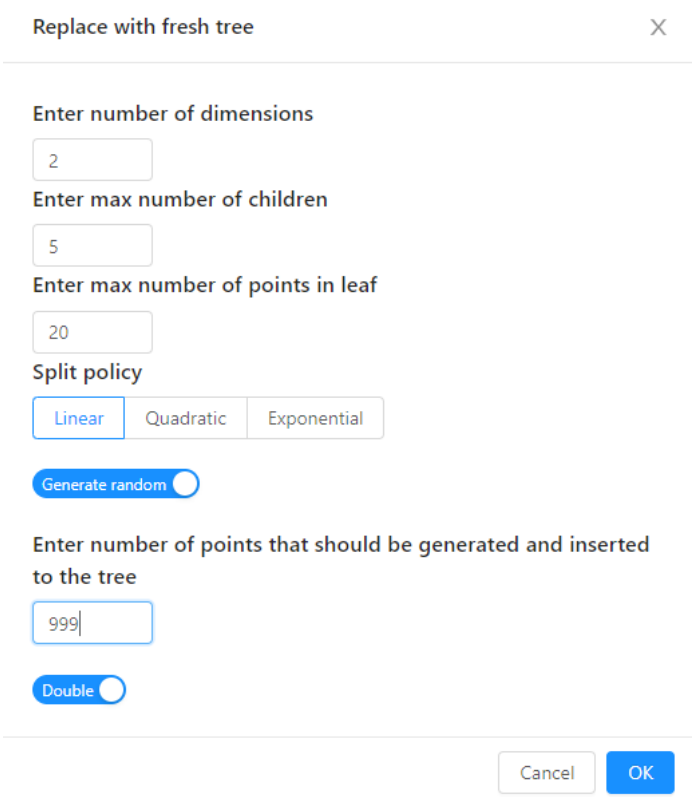

Figure 8: Záložka Data, formulář Replace with fresh

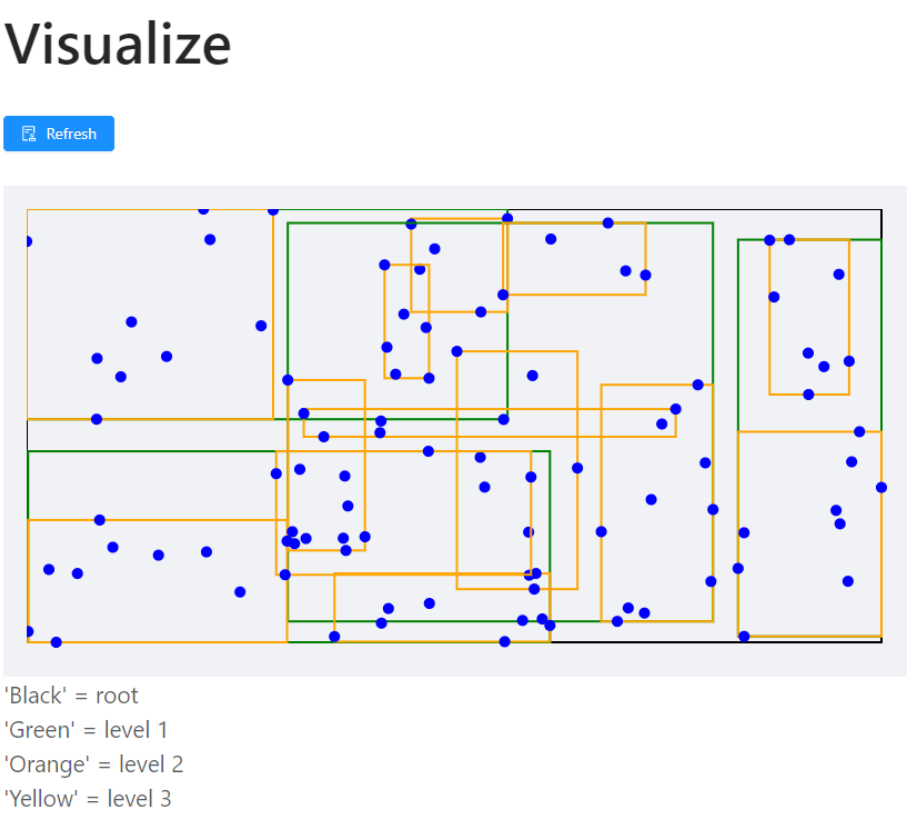

Figure 9: Záložka Visualize

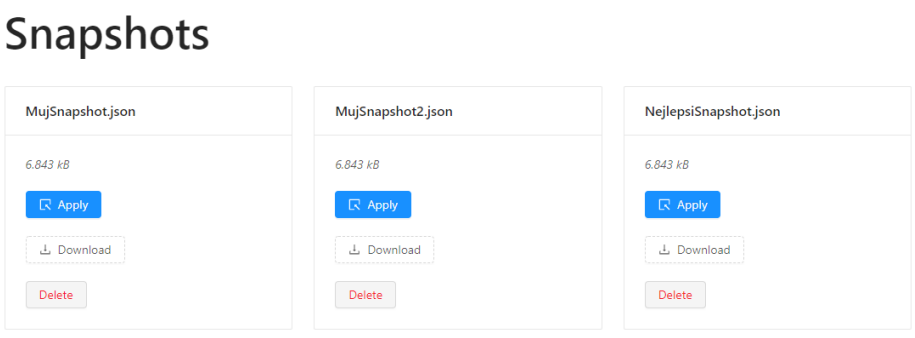

Figure 10: Záložka Snapshots

5 Experimentální sekce

Při vytváření R-stromu můžeme nastavit několik parametrů.

- Number of dimensions
- Max number of children
- Max number of points in leaf
- Split policy

S těmito parametry je možné pohybovat a sledovat rychlost vytvoření stromu nebo rychlost dotazů nad stromem v závislosti na těchto parametrech. Dále můžeme porovnávat rychlost dotazů nad stromem s rychlostí dotazů provedených nad stejnými daty, ale lineárně.

5.1 Rychlost vkládání bodů

V tomto testu jsme vygenerovali 100 000 náhodných bodů, které jsme vložili do stromu a měřili rychlost této operace v závislosti na parametrech Max number of children a Max number of points in leaf, které jsou oba nastaveny na stejnou hodnotu, uvedenou na ose x. Parametr Number of dimensions byl nastaven na 3. Test jsme provedli desetkrát a výsledky zprůměrovali pro vyšší přesnost měření.

V grafu můžeme vidět, že zatímco Linear a Quadratic algoritmy se zvyšujícím se počtem Max children běží pořád téměř stejnou dobu, Exponential algoritmus se podle očekávání výrazně zpomaluje. Obrázek 11

5.2 Rychlost dotazování

V testu rychlosti dotazů jsme nastavili Number of dimensions na 2 a Max number of children na 15. Poté jsme náhodně vygenerovali 2 000 000 bodů, které byly vloženy do stromu. Na tomto stromě jsme provedli 1000 náhodných dotazů. Range query - vybrán náhodný bod a náhodná range, KNN query - vybrá náhodný bod a hledáme 100 nejbližších sousedů.

Každý dotaz byl proveden tisíckrát a časy byly sečteny pro každé spuštění. Měřili jsme čas vytvoření datové struktury, čas běhu rozsahového dotazu a čas běhu KNN dotazu.

Lineární vyhledávání - vytvoření datové struktury pro lineární vyhledávání je velice rychlé, naopak dotazy jsou v porovnání s R-stromem pomalé a to zejména KNN dotaz

R-strom - Linear split - vytvoření stromu trvá déle, nicméně na rychlosti dotazů můžeme pozorovat výrazné zlepšení

R-strom - Quadratic split - vytváření stromu trvá nejdéle, ale na druhou stranu strom dosahuje nejlepších výsledků při porovnání rychlosti dotazů

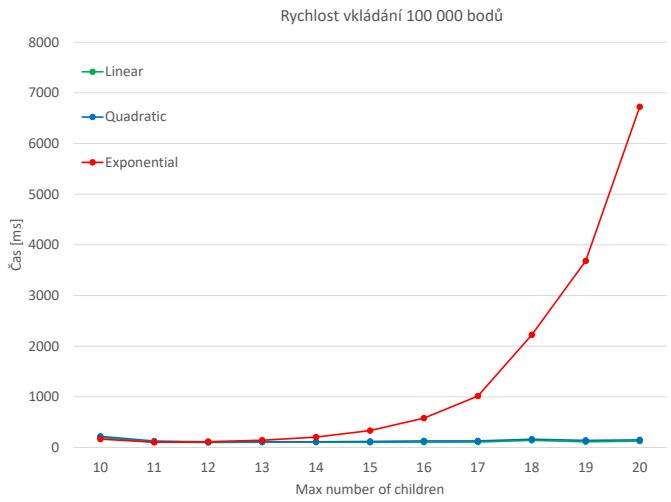

Figure 11: Graf rychlosti vkládání při použití různých Split policy

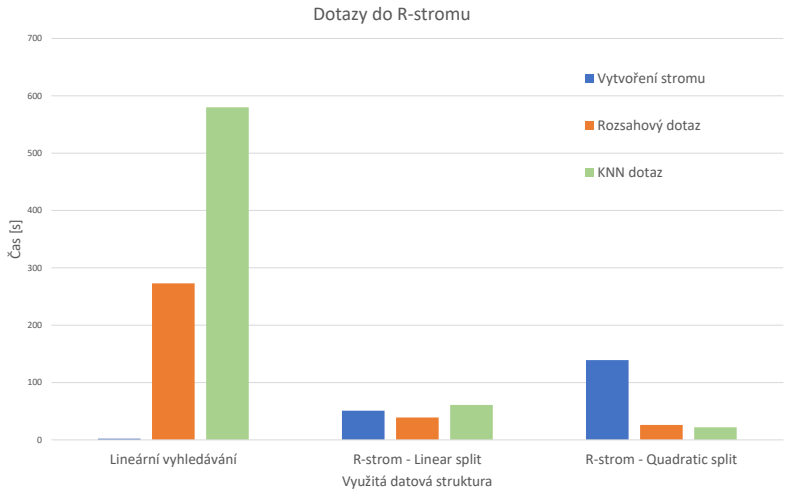

Figure 12: Graf rychlosti vytvoření stromu a různých dotazů

6 Diskuze

Experimenty dopadly podle očekávání. Vkládání nových bodů do stromu s využitím exponenciálního algoritmu pro dělení uzlů se ukázalo jako velmi pomalé. Předpokládáme tedy, že největší využití bude mít algoritmus kvadratický.

Linear split i Quadratic split algoritmy dosahují velmi dobrých výsledků v porovnání s lineárním vyhledáváním. Největší zlepšení je vidět při provádění KNN dotazů, které je u Quadratic splitu zhruba 3x rychlejší, než u Linear splitu a dokonce 25x rychlejší, než lineární vyhledávání.

Možná vylepšení - do R-stromu bychom mohli přidat i operaci mazání prvku. S touto operací souvisí i operace snížení počtu hladin stromu, když počet prvků v uzlu klesne pod určitou mez.

7 Závěr

Cílem bylo vytvořit R-strom, což se nám podařilo. Díky vytvořenému API je R-strom možné snadno použít pro nějaká reálná data. Mělo by tedy být relativně snadné použít tuto implementaci v některém reálném projektu, který pracuje s body v prostoru. Při rozsahových dotazech i KNN dotazech je strom několikanásobně rychlejší, než lineární vyhledávání a obecně strom i webová aplikace vracejí výsledky správně a rychle.

References

[1] *MANOLOPOULOS, Yannis. R-trees: theory and applications.* London: Springer, c2006. Advanced information and knowledge processing. ISBN 18-523-3977-2.

[2] *Nearest Neighbour Queries* [online]. [cit. 2019-05-07]. Dostupné z: `https://infolab.usc.edu/csci599/Fall2009/slides/Nearest%20Neighbor%20Queries%20Slides.pdf`

[3] *Json.NET - Newtonsoft* [online]. [cit. 2019-05-07]. Dostupné z: `https://www.newtonsoft.com/json`

[4] *NETStandard.Library* [online]. [cit. 2019-05-07]. Dostupné z: `https://www.nuget.org/packages/NETStandard.Library/`

[5] *Microsoft.AspNetCore.All* [online]. [cit. 2019-05-07]. Dostupné z: `https://www.nuget.org/packages/Microsoft.AspNetCore.All/`

[6] *Microsoft.NETCore.App* [online]. [cit. 2019-05-07]. Dostupné z: `https://www.nuget.org/packages/Microsoft.NETCore.App`

[7] *Microsoft.Extensions.PlatformAbstractions* [online]. [cit. 2019-05-07]. Dostupné z: `https://www.nuget.org/packages/Microsoft.Extensions.PlatformAbstractions/`

[8] *Swashbuckle.AspNetCore* [online]. [cit. 2019-05-07]. Dostupné z: `https://www.nuget.org/packages/Swashbuckle.AspNetCore/5.0.0-rc2`

[9] *React* [online]. [cit. 2019-05-07]. Dostupné z: `https://reactjs.org/`

[10] *Ant Design* [online]. [cit. 2019-05-07]. Dostupné z: `https://ant.design/`

[11] *Konva* [online]. [cit. 2019-05-07]. Dostupné z: `https://konvajs.org/`

[12] *Download .NET* [online]. [cit. 2019-05-08]. Dostupné z: `https://dotnet.microsoft.com/download`

[13] *Node.js* [online]. [cit. 2019-05-08]. Dostupné z: `https://nodejs.org/en/`

[14] *Node package manager* [online]. [cit. 2019-05-08]. Dostupné z: `https://www.npmjs.com/`