

# M-strom

Jan Jeníček, Michal Šolc

## Popis projektu

Cílem tohoto projektu je implementace metrické přístupové metody (MAM) M-strom. M-strom je stromová struktura využívající axiomů metrického prostoru (nezápornost, totožnost, symetrie, trojúhelníková nerovnost) pro zrychlení operací nad datovou sadou. Poloha objektů v metrickém prostoru je definována pouze vzdáleností od ostatních objektů. Jelikož je M-strom metrickou metodou, mohou být jako data chápány jakékoliv objekty, pro které je definována metrika, neboli vzdálenostní funkce splňující zmiňované axiomy. Samotná implementace je tedy generická a umožňuje využití jakékoliv datové sady, pokud je M-stromu definována vzdálenostní funkce nad těmito daty.

Implementace rozšiřuje zadání a vyjma rozsahových a KNN dotazů umožňuje i dynamické přidávání a odebrání dat z M-stromu. Umožňuje také výběr libovolné heuristiky při dělení jednotlivých uzlů stromu v případě přetečení jejich maximální definované kapacity.

Pro prezentaci funkčnosti této implementace je vytvořena jednoduchá GUI aplikace, která pracuje s 3D celočíselnými vektory a Eukleidovskou vzdáleností jako metrikou. Je možné začít s prázdnou datovou sadou, načíst datovou sadu z předchozího běhu aplikace, nebo nechat vygenerovat náhodnou datovou sadu.

GUI aplikace také obsahuje testovací modul, který umožňuje generovat náhodná data a dotazy. Ty slouží k otestování rychlosti vkládání nových dat a dotazování se nad M-stromem při použití různých dělicích heuristik.

## Způsob řešení

M-strom je zakořeněný strom a všechny definované operace jsou implementovány pro jeho jednotlivé uzly. Uzel je z abstraktního pohledu reprezentován metrickým regionem definovaným středem a poloměrem. Každý region dále obsahuje všechny podstromy daného uzlu, které jsou taktéž reprezentovány jako metrické regiony. Výjimkou jsou pouze listy stromu, které již nemají žádné podstromy a obsahují pouze množinu bodů v metrickém prostoru (dat). Z implementačního pohledu každý uzel obsahuje data, která definují jeho polohu v prostoru, poloměr metrického regionu (hyperkoule), jímž je reprezentován a množinu odkazů na jednotlivé podstromy uzlu. Listy pak nemají množinu odkazů na podstromy, ale na samotná data. Tato implementace jednotlivé operace vždy aplikuje pouze na kořen, který danou operaci vždy propaguje dál směrem k listům.

Operace definované nad M-stromem:

- Vkládání dat
- Mazání dat
- Rozsahové dotazy – vrací všechny objekty s maximální mírou odlišnosti (vzdálenosti)
- KNN dotazy – vrací ,k' nejpodobnějších (nejbližších) objektů

### Vkládání dat

Začneme vložením dat do kořene. Pokud se nenacházíme v listu, nalezneme nejvhodnější podstrom pro vkládaná data a vložíme je do něj. Při hledání vhodného podstromu mohou nastat 2 situace:

- Vkládaná data se vejdou do některého z podstromů
- Vkládaná data se nevejdou do žádného z podstromů

V prvním případě zvolíme jako nejvhodnější takový podstrom, do něhož se data vejdou a jsou nejbliž k jeho středu. V druhém případě zvolíme podstrom, který je po vložení dat třeba nejméně expandovat. Pokud se nacházíme v listu, data do něho jednoduše vložíme.

Po vložení je třeba zkontrolovat, zda nepřetekla kapacita některého uzlu. Zde začínáme naopak v listu a informaci propagujeme směrem ke kořeni. V případě, že kapacita uzlu přetekla, rozdělíme ho na dva nové uzly za použití definované heuristiky. Tím se zvětší počet podstromů rodičovského uzlu a může tak nastat jeho dělení.

## Mazání dat

Začneme v kořeni stromu. V něm se data pokusíme vymazat ze všech podstromů, které by jej mohly obsahovat. Dále propagujeme kromě dat také informaci o vzdálenosti dat od rodičovského uzlu. Ve vnitřních uzlech nejprve pomocí vzdálenosti dat od rodiče a trojúhelníkové nerovnosti odfiltrujeme podstromy, v nichž se data nacházet nemohou (čímž ušetříme počítání vzdálenostní funkce mezi daty a vyfiltrovanými podstromy). Poté se analogicky s kořenem pokusíme smazat data ze všech podstromů, v nichž by se mohly nacházet. Pokud se nacházíme v listu a data jsou zde skutečně uložena, smažeme je.

Po mazání se může stát, že se počet objektů v uzlu dostane pod jeho minimální kapacitu. Tato situace není implementována, neboť se v ukázkové aplikaci nepředpokládá časté mazání dat. V opačném případě by se hledal dárcovský uzel.

## Rozsahové dotazy

Vstupem pro rozsahový dotaz jsou data a maximální míra odlišnosti, tedy vzdálenost – rozsahový dotaz je tak reprezentován metrickým regionem (hyperkoulí), stejně jako jednotlivé uzly stromu. Začneme v kořeni a pokusíme se hledat ve všech jeho podstromech, které jsou s dotazem v průniku. Spolu s dotazem propagujeme i vzdálenost dat k rodičovskému uzlu. Ta je ve vnitřních uzlech opět využívána k odfiltrování některých podstromů za použití trojúhelníkové nerovnosti a tím je snížen počet volání vzdálenostní funkce. Pokud se nacházíme v listu, vrátíme všechna data, která jsou v průniku s metrickým regionem dotazu.

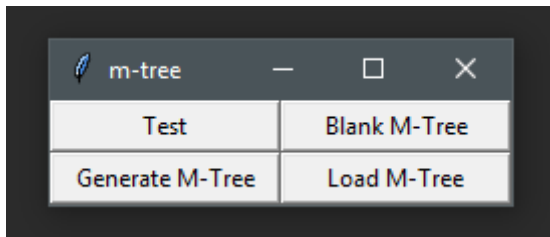
## KNN dotazy

Jsou implementovány obdobně, jako rozsahové dotazy. Rozdílem je, že za metrický region dotazu je považován celý metrický prostor a z listu je vráceno maximálně  $k$  nejlepších objektů (ty jsou nejprve seřazeny). V nelistových uzlech je pak vráceno maximálně  $k$  nejlepších objektů ze všech podstromů uzlu. K vybrání těchto  $k$  nejlepších objektů je použit podobný princip, jako u např. merge sortu.

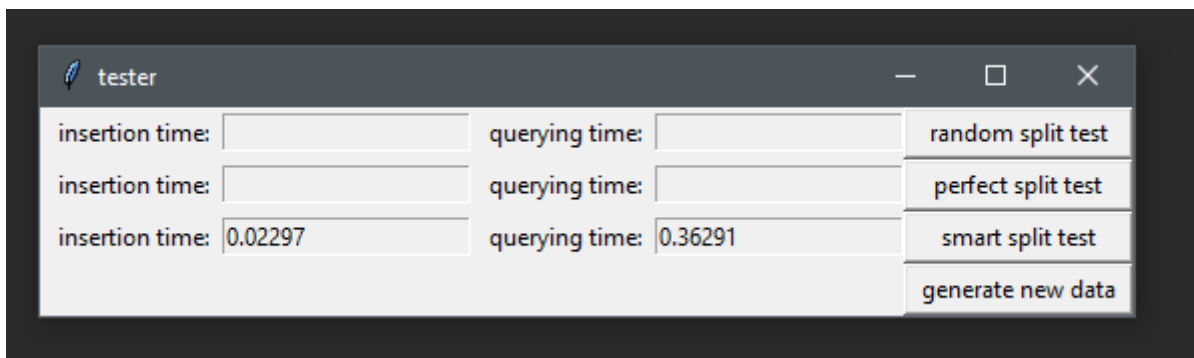
## Implementace

Celý projekt je napsaný v jazyce Python (konkrétně používá Python 3.7). Jedná se o konzolovou aplikaci, nad kterou je dle zadání postaveno jednoduché GUI. To je implementováno za použití pro Python vestavěné knihovny tkinter. Samotný M-strom je generický a je zabalen do samostatného balíčku mtree a je samostatně použitelný v jakékoliv jiné aplikaci. V aplikaci nejsou použity žádné externí knihovny či nástroje vyžadující instalaci. Pro spuštění aplikace tak stačí spustit Python skript main.py. GUI aplikace sloužící k demonstraci funkcí M-stromu pracuje s 3D celočíselnými vektory. Data je třeba zadávat jako 3 celá čísla oddělená mezerou.

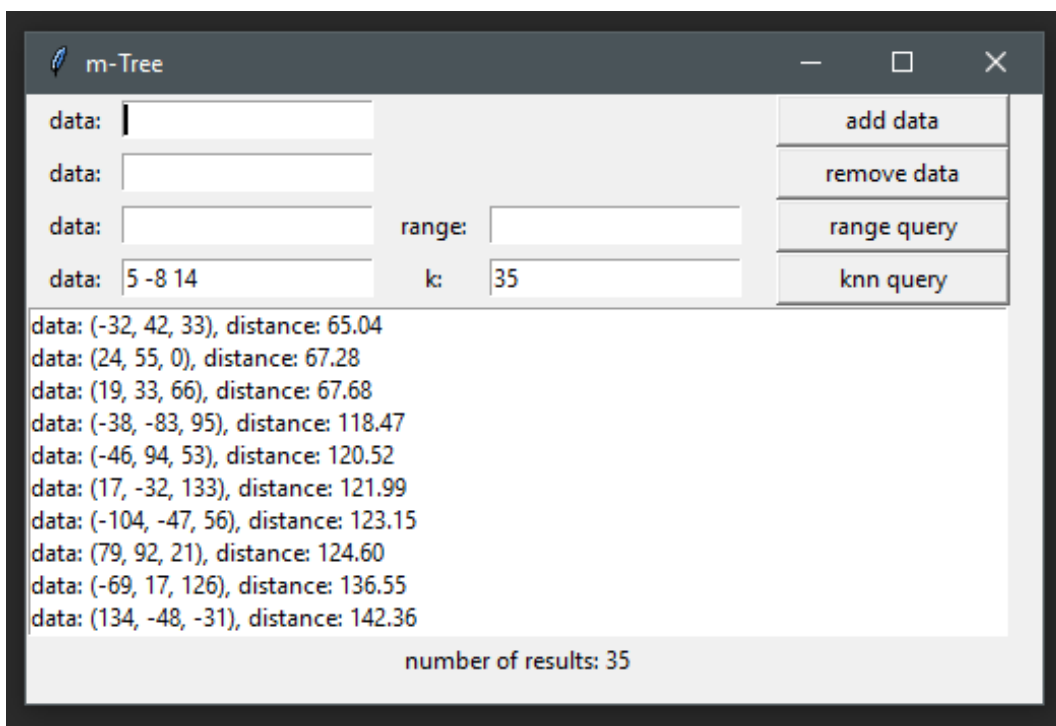
## Příklad výstupu



Hlavní menu



Tester



Rozhraní pro M-strom (zde KNN dotaz nad vygenerovaným stromem)

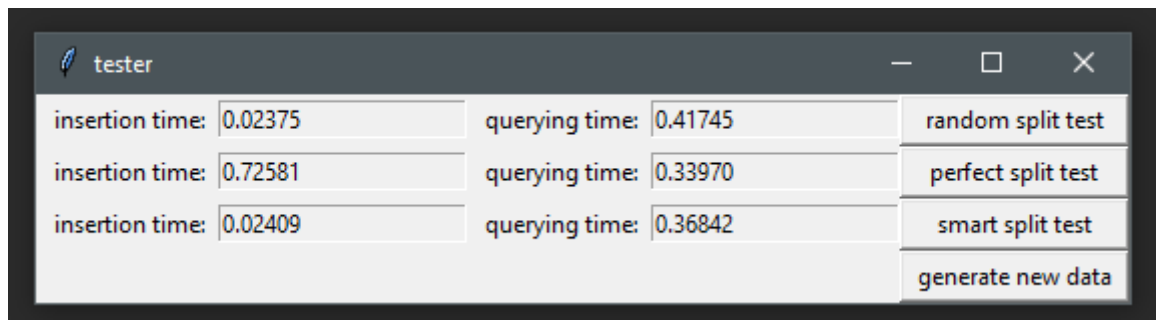
## Experimentální sekce

V této sekci jsme se rozhodli zaměřit na vliv dělicí heuristické funkce na rychlost vkládání dat do stromu a na následnou rychlost dotazování se nad tímto stromem. Při experimentu jsou využity tři různé dělicí funkce:

- Náhodné dělení – zvolí dva náhodné objekty jako středy nových uzlů. Následně zamíchá daty a rozdělí je na poloviny. Každou z polovin poté přiřadí jinému ze vzniklých uzlů.
- Perfektní dělení – vyzkouší všechny možné kombinace, jak lze uzel rozdělit a pro každou z kombinací vyzkouší všechny možné středy. Dělení je tím lepší, čím menší průnik spolu mají dvě hyperkoule reprezentující metrické regiony nově vzniklých uzlů. Funkce vybere nejlepší možné rozdělení s nejlepším možným středem.
- Chytré dělení – jako nové středy zvolí dva objekty daleko od sebe a zbylé objekty přiřazuje vždy tomu středu, který je blíž.

### Výsledky experimentů

Experiment dopadl přesně podle očekávání. Náhodná heuristika má lineární časovou složitost (implementovaná heuristika míchá daty, pokud by se tak nedělo, dosahovala by konstantní časové složitosti) a dělení uzlů je tak velmi rychlé, nebere ale v potaz následnou strukturu stromu. Díky tomu dosahuje nejlepších výsledků při vkládání, ale velmi špatných výsledků při dotazování, jelikož dotazovací algoritmus musí projít mnohem více uzlů stromu. Perfektní dělení dosahuje exponenciální časové složitosti a vkládání dat je tak extrémně náročné, zato ale vždy nalezne ten nejlepší možný způsob, jak uzel rozdělit a dotazování se nad tímto stromem je nejrychlejší možné. Chytré dělení je kombinací dvou předchozích heuristik. Přestože dělení uzlů není náhodné, zachovává si heuristika lineární časovou složitost a dosahuje tak při vkládání velmi podobných časů, jako dělení náhodné. Zároveň je strom uspořádaný, takže dotazování je podobně rychlé, jako při použití perfektní heuristiky.



insertion time:	0.02375	querying time:	0.41745	random split test
insertion time:	0.72581	querying time:	0.33970	perfect split test
insertion time:	0.02409	querying time:	0.36842	smart split test
				generate new data

Obrázek ukazuje výsledek testů nad jednoduchými 3D vektory. Testování proběhlo nad 10 různými datovými sadami a výsledek je zprůměrovaný. V každém z testů bylo přidáno 50 záznamů a následně provedeno 20 rozsahových a 20 KNN dotazů. Heuristiky se chovali obdobně i s vzrůstajícím počtu dimenzí dat. Rozdíly (především při dotazování) byly pouze markantnější. To je pravděpodobně způsobeno náročnější vzdálenostní funkcí.

## Diskuze

Aplikace obsahuje několik nedokonalostí. Ačkoliv implementace podporuje mazání dat z M-stromu, neřeší balancování uzlů, které se dostanou pod minimální kapacitu, jelikož ukázková aplikace nepředpokládá mazání velkého množství dat z databáze. Implementace je sice perzistentní a umožňuje pracování s daty z minulého běhu, používá však k ukládání dat jen jednoduchý Python modul pickle, který slouží k uložení celého M-stromu. Je tedy při každém běhu třeba načíst celý index do paměti. Prostor je také pro zlepšení celkového chování M-stromu, pro nějž za dobu jeho existence byly vytvořeny různé modifikace, které optimalizují nad ním definované operace. Současně aplikace prozatím obsahuje jen velmi jednoduché GUI. Při testování je například nutné čekat, než test doběhne a aplikace během něho není interaktivní.

## Závěr

Experimenty ukázaly důležitost výběru optimální heuristické funkce pro dělení uzlů. Příliš špatné heuristické funkce způsobují špatné výsledky při dotazování, zatímco početně náročné funkce jsou nepoužitelné kvůli velmi pomalému ukládání. Časové testy také stvrdili, že i základní verze M-stromu může být efektivní metoda pro podobnostní vyhledávání jakýchkoliv multimediálních objektů, pro něž je možné definovat metriku.