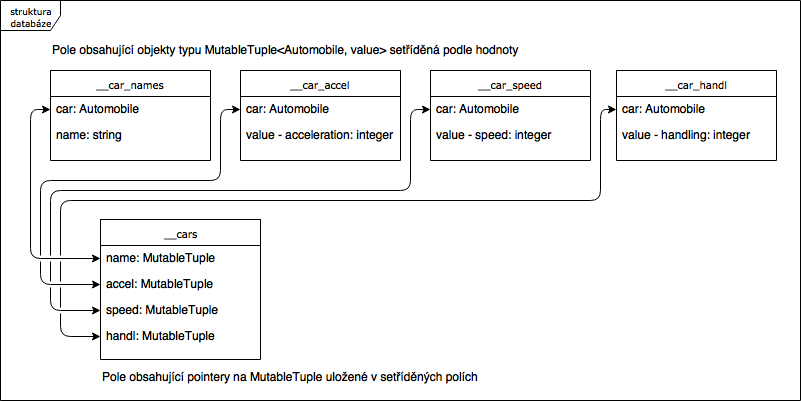
Threshold top - k algoritmus  
Vyhledávání top-k (Threshold) výsledků v DB libovolných produktů s možností volby agregační funkce + GUI

Popis projektu  
Implementoval jsem jednoduchu webovou aplikaci pro demonstraci použití treshold top(k) algoritmu v jazyce Python. Mé řešení obsahuje navíc i naivní algoritmus a faginův algoritmus. Je tak možné přehledně vyzkoušet, jak jednotlivé algoritmy fungují.

Způsob řešení

Jako databázi využívám čtyři pole obsahující KeyValuePair ve tvaru <pointer na třídu Automobile, numerická hodnota> . Tyto pole jsou na základě hodnoty setříděny při načtení databáze ze souboru. Navíc mám pole obsahující třídu Automobile, v níž si držím pointery na jednotlivé položky v polích obsahujících hodnoty.



Samotný top-k treshold využívá minimovou haldu k uchovávání objektů a efektivní odstraňování prvků s nejnižším agregačním score.

## Samotný algoritmus funguje následujícím způsobem:

1. Nastavím treshold t jako agregační hodnotu prvků zjištěných v přístupu do databáze.
2. Náhodným přístupem vypočítám skóre nalezených objektů.
3. V minimové haldě si uchovávám seznam top-k objektů které jsem doposud zobrazil.
4. Zastavím se, pokud jsou skóre top-k objeků v haldě větší nebo rovny tresholdu.  
   Konec – vrátím top-k objektů které jsem prozkoumal.

Implementace

Aplikace je napsaná v programovacím jazyce Python s využitím webového frameworku FLASK pro generování webové stránky. Samotný web je HTML dokument s několika funkcemi psanými v jazyce JavaScript.

# Příklad výstupu

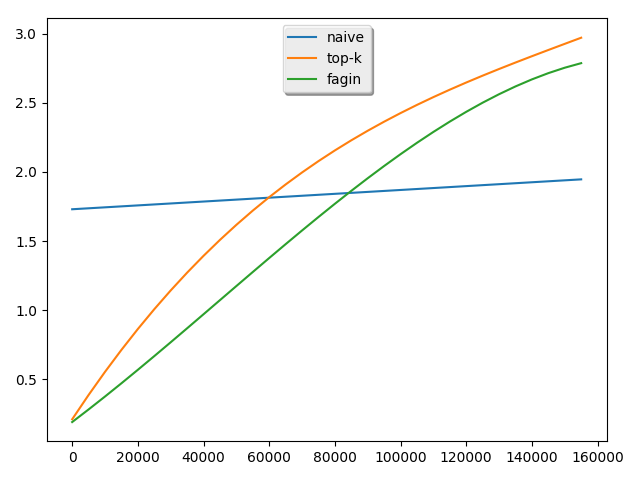
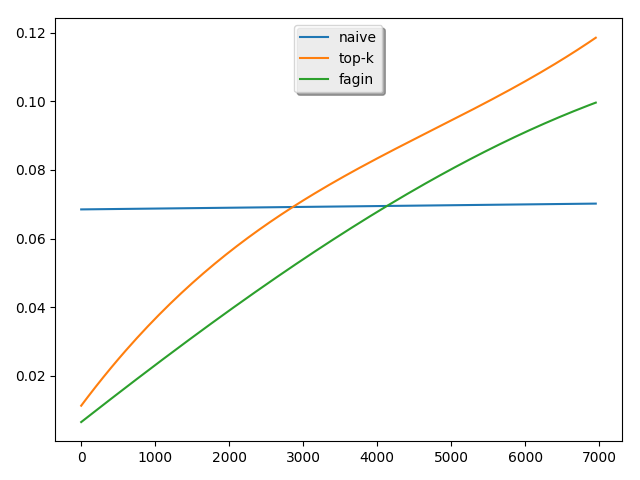
Na screenshotu vidíme ukázkovou query nad databází aut ze hry Need for Speed: Most Wanted. Na query byla využita jako agregační funkce suma a byl využit top-k treshold algoritmus. Pod vypsanou tabulkou přehledně vidíme, že query trvala 0.0003s a využila 17 přístupů do databáze.

# 

# Experimentální sekce

V následující sekci se pokusím porovnat faginův a top-k algoritmus na základě výpočetního času i počtu přístupů do databáze.

## Porovnání doby běhu



Srovnání doby běhu algoritmů na databázi o velikosti 7 000 a 160 000 objektů. Je vidět, že ačkoliv použije algoritmus top-k nižší množství náhodných přístupů, vyžaduje udržování minimální haldy pro uchování top-k elementů delší výpočetní čas. Pokud mám tedy rychlý přístup k databázi s náhodným přístupem, vyplatí se využít faginův algoritmus.

## 

## Porovnání počtu přístupů do databáze

Je vidět, že algoritmus top-k využívá ve srovnání s faginovým algoritmem značně menší počet přístupů do databáze a proto v prostředí, s pomalým přístupem k databázi se i za cenu větší výpočetních nároků vyplatí.

* Většina projektů lze posuzovat z hlediska přesnosti či rychlosti (nebo obojího), přičemž tyto jsou závislé na různých vstupních parametrech projektu. V této sekci by měly být takové parametry zkoumány. Např. rychlost typicky závisí na velikosti vstupu nebo naopak velikosti výstupu. Lze pak například do grafu nebo tabulku vynést takovéto závislosti.

Diskuze  
- Většina projektů je typu “proof of koncept“, tj. jde o vyzkoušení poznatků prezentovaných v přednáškách v praxi. Nejde tedy o detailní řešení všech problémů, které mohou při implementaci nastat – takový projekt by dalece přesahoval rámec semestrálního projektu. Tato sekce tedy obsahuje rozbor těchto nedostatků a potenciálním způsobu jejich řešení.

Závěr

Ačkoliv jsem toto zadání dostal přidělené   
- Stručné shrnutí toho, co se řešilo, zhodnocení výsledků.