**ŽILINSKÁ UNIVERZITA V ŽILINE**

**FAKULTA RIADENIA A INFORMATIKY**

Algoritmy a údajové štruktúry 1

Domáce zadanie 3

Analýza výkonu tabuliek

Adam Parimucha

5ZYI27

2020/2021

# UML diagram

Trieda **TableTester**

atribúty:

* stopky typu TimeMeasurement, čo je trieda, ktorú používam na meranie času jednotlivých operácií
* scenar typu char - písmeno vykonávaného scenára
* pocetOp typu int - počet operácií vykonaných v príslušnom nastavenom scenári
* podielVloz,podielZrus,podielSpristupni typu int – podiely jednotlivých operácií v nastavenom scenári

metódy:

konštruktor – priradí atribútu stopky novú inštanciu triedy Timemeasurement

deštruktor – zmaže stopky

spustiTesty – berie ako parameter tabulku, na ktorej sa vykonávajú testy, názov CSV súboru, do ktorého sa zapisujú časy vykonávania operácií pre jednotlivé scenáre, písmeno scenára, ktorý sa má vykonať a počet operácií a podiel operácií vlož, sprístupni a zruš. Počet operácií a ich podiely sú nepovinné, defaultne sú nastavené na 0. Popis realizácie priebehu jednotlivých scenárov v testoch je popísaný nižšie.

vloz – metóda spustí stopky, vykoná vloženie údajov s kľúčom a dátami zadanými v parametri do tabuľky zadanej v parametri a zastaví stopky

odstran – metóda spustí stopky, vykoná odstránenie údajov s kľúčom zadaným v parametri z tabuľky zadanej v parametri a zastaví stopky

spristupni – metóda spustí stopky, vykoná sprístupnenie prvku s kľúčom zadaným v parametri z tabuľky zadanej v parametri a zastaví stopky

nastavScenar – metóda priradí do príslušných atribútov počet operácií a podiely operácií vlož,sprístupni a odstráň podľa scenára, ktorého písmeno je zadané v parametri z konštánt definovaných v triede.

zapis – zapíše do ofstreamu zadaného v parametri dáta vo formáte nazovOperacie;pocetPrvkovVTabulke;casVykonaniaOperacie\n a resetne stopky.

Trieda **TestApp**

metódy:

spustiVyberTabulky – v metóde sa vytvorí pointer typu Table<K,T> a následne sa aplikácia užívateľa spýta, ktorú tabuľku chce testovať. Podľa voľby užívateľa sa do pointra priradí príslušný typ tabuľky a nad tabuľkou sa postupne zavolá metóda spustiTesty pre scenáre A,B a C

# Popis realizácie priebehu jednotlivých scenárov v testoch

Na začiatku testu sa tabuľka vyčistí a následne prebehne počet operácií definovaných v scenári, čo je v našich scenároch A,B,C 100000. Pred každou operáciou sa vygeneruje náhodné číslo od 1 do 100 a náhodný kľúč a operácia sa vyberie nasledovným spôsobom: ak číslo padne do intervalu <1,podiel vložení>, vykoná sa operácia Vlož, ak do intervalu (podiel vložení, podiel sprístupnení>, vykoná sa operáci sprístupni a v ostatných prípadoch sa vykoná operácia odstráň. Pri vkladaní sa vytvorí prázdny objekt typu T, aby bola zaručená univerzálnosť. Je tiež ošetrené, aby sa neodstraňoval alebo nesprístupňoval prvok, ktorého kľúč nie je v tabuľke, lebo inak by mohol program spadnúť s chybou. Podiely operácií pre jednotlivé scenáre možno vidieť v nasledovnej tabuľke:

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenár** | **Podiel operácií** | | |
| **Vlož na základe kľúča** | **Sprístupni na základe kľúča** | **Odstráň na základe kľúča** |
| **A** | 30 | 60 | 10 |
| **B** | 45 | 10 | 45 |
| **C** | 30 | 40 | 30 |

# Popis formátu údajov a CSV súborov v testoch

Prvý stĺpec v CSV súbore je názov operácie, v druhom stĺpci je aktuálny počet prvkov pri vykonaní operácie a v treťom čas v nanosekundách, za ktorý sa operácia vykonala.

# Analýza výsledných CSV súborov

## Preskúmanie zložitostí jednotlivých operácií v závislosti od veľkosti testovanej štruktúry

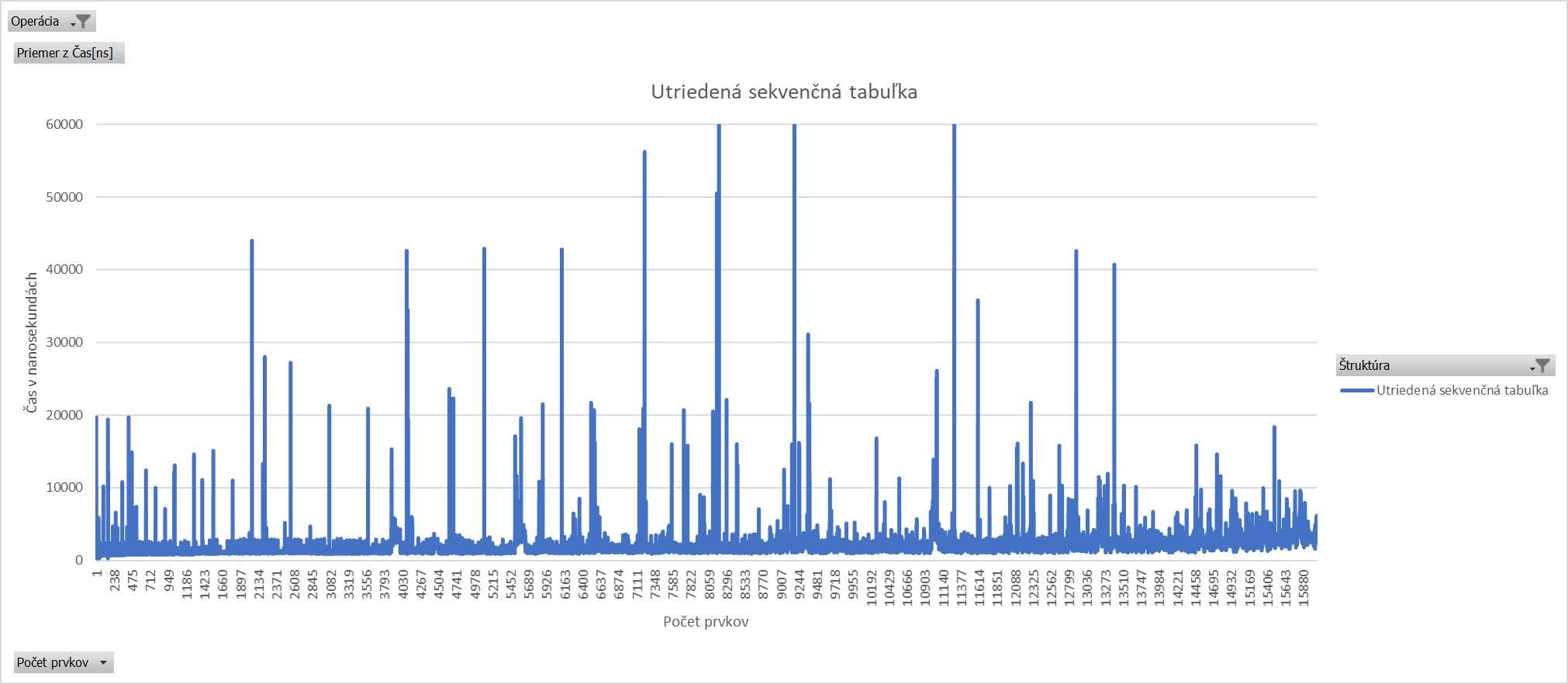
Pre preskúmanie zložitostí jednotlivých operácií som vykonal scenár C na každej z tabuliek a následne som výsledné súbory spojil do jedného a vytvoril z nich kontingenčnú tabuľku a graf. V grafe je na osi x počet prvkov a na osi y priemerný čas vykonanie operácie pre príslušný počet prvkov v nanosekundách.

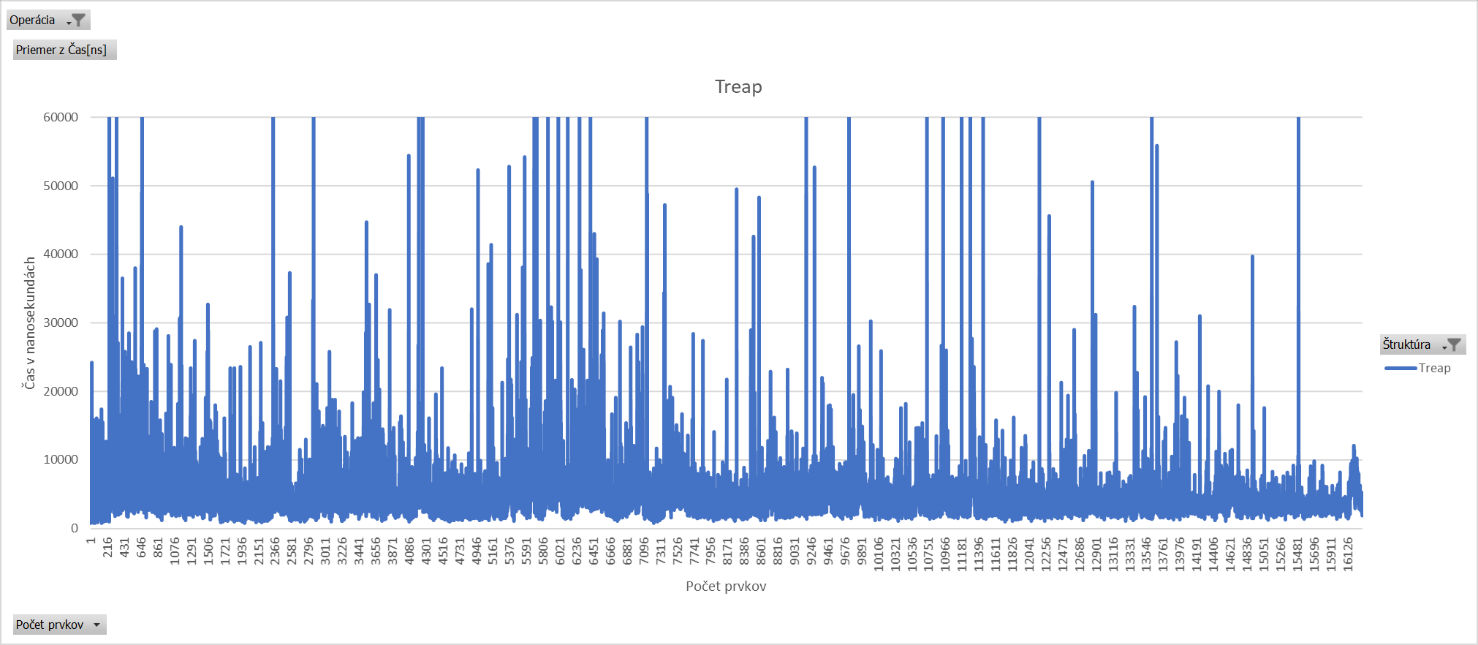
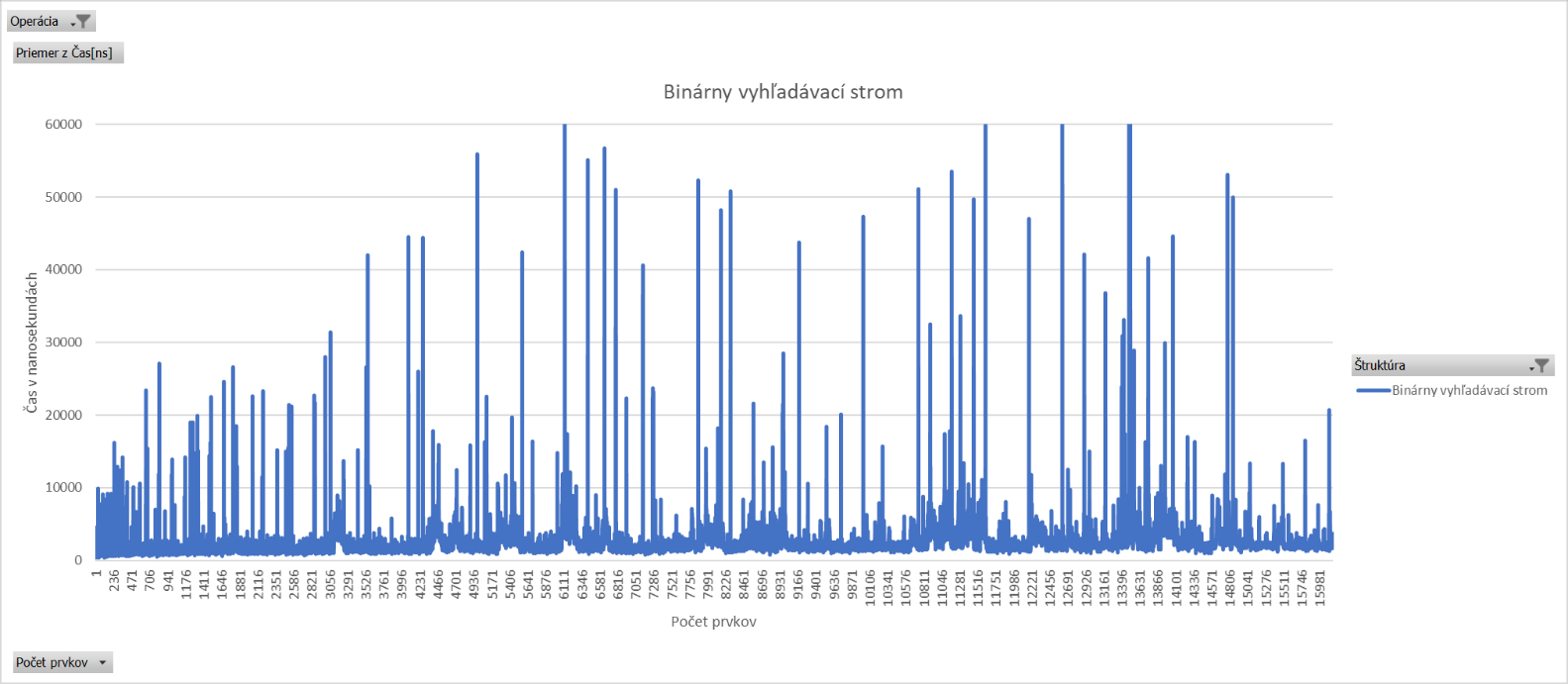
Tabuľka teoretických zložitostí testovaných štruktúr:

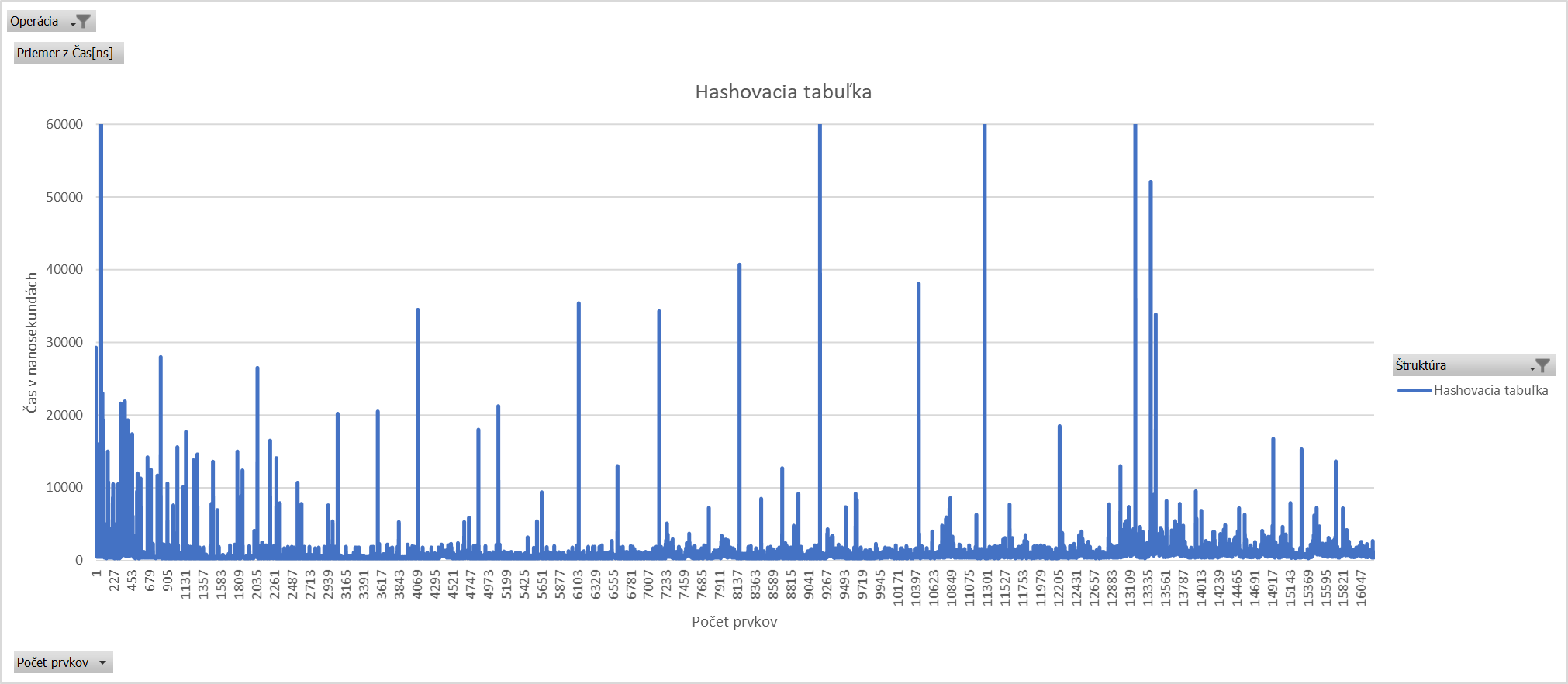
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **štruktúra** | | **operácia** | | |
| **Vlož prvok** | **Zruš prvok\*** | **Nájdi prvok** |
| **Sekvenčná utriedená tabuľka** | | O(N) | O(N) | O(log N) |
| **Binárny vyhľadávací strom** | Ideálny prípad | O(log N) | O(log N) | O(log N) |
| Najhorší prípad | O(N) | O(N) | O(N) |
| **Treap** | | O(log N) | O(log N) | O(log N) |
| **Hashovacia tabuľka** | | O(1) | O(1) | O(1) |
| \*Operácia zruš nezahŕňa vyhľadanie prvku | |  |  |  |

**Namerané grafy zložitostí:**

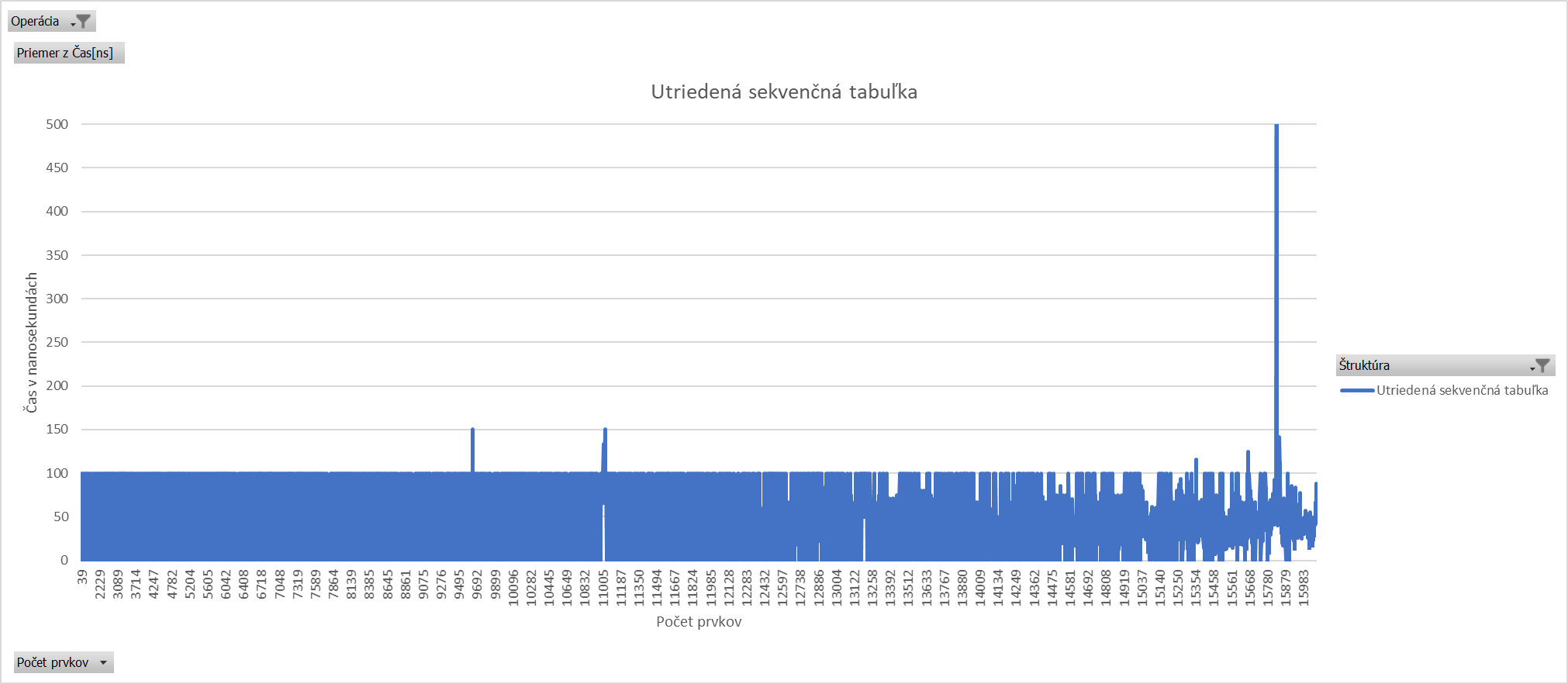
**Operácia Vlož prvok na základe kľúča**

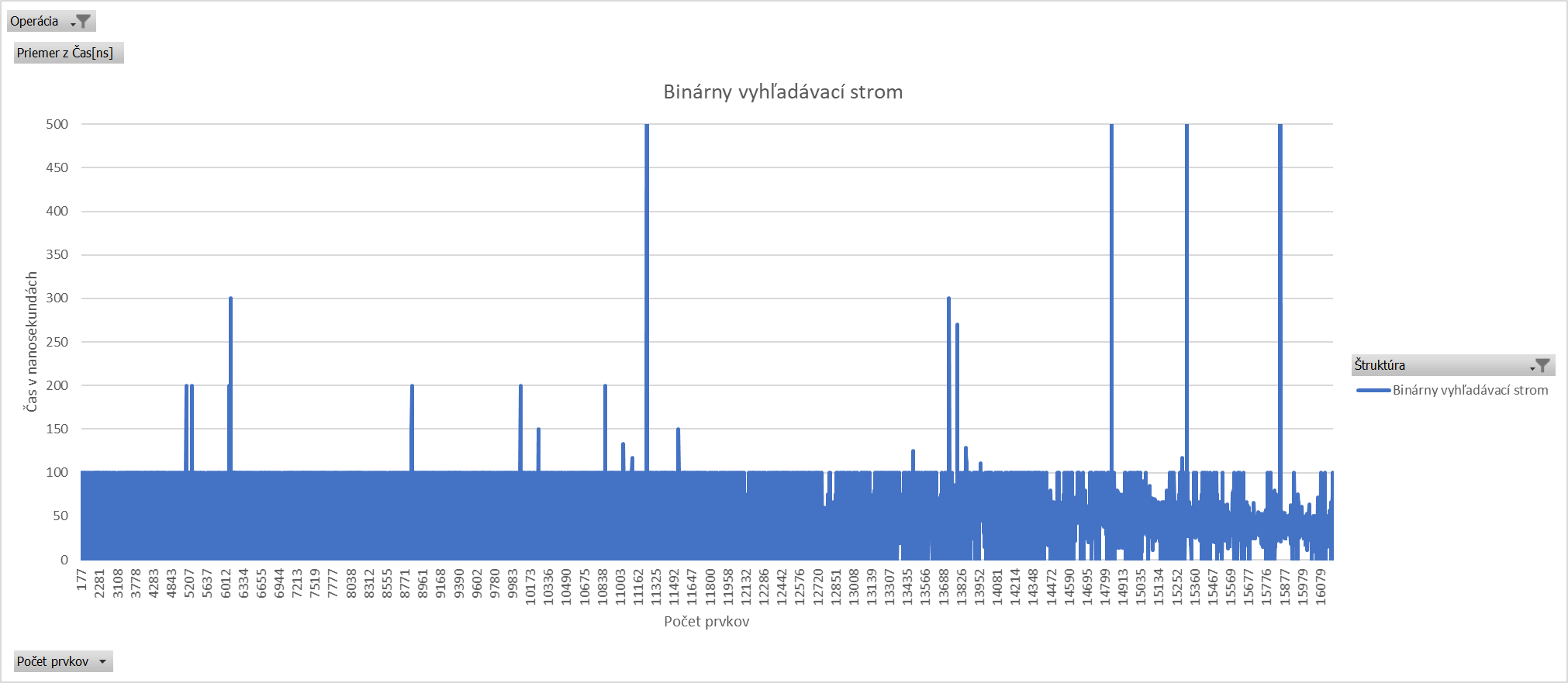


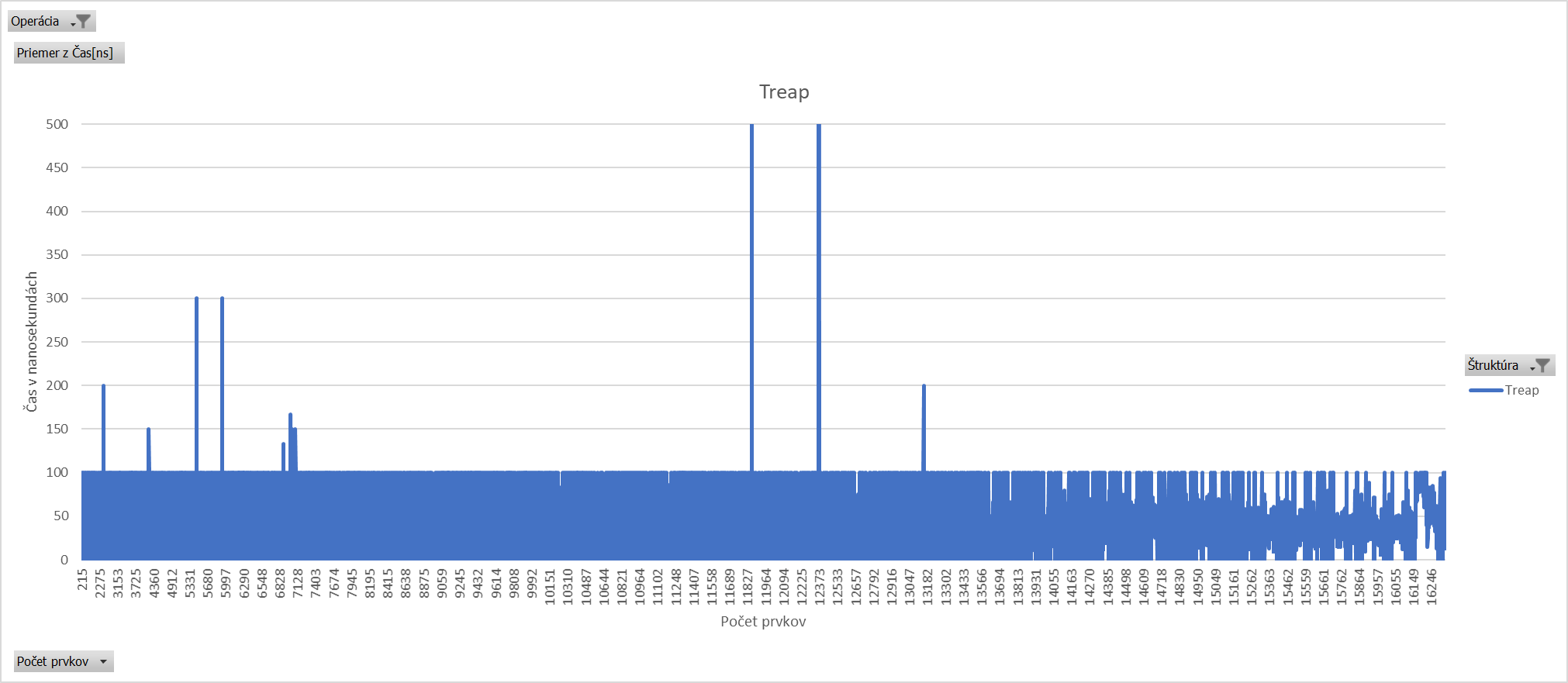


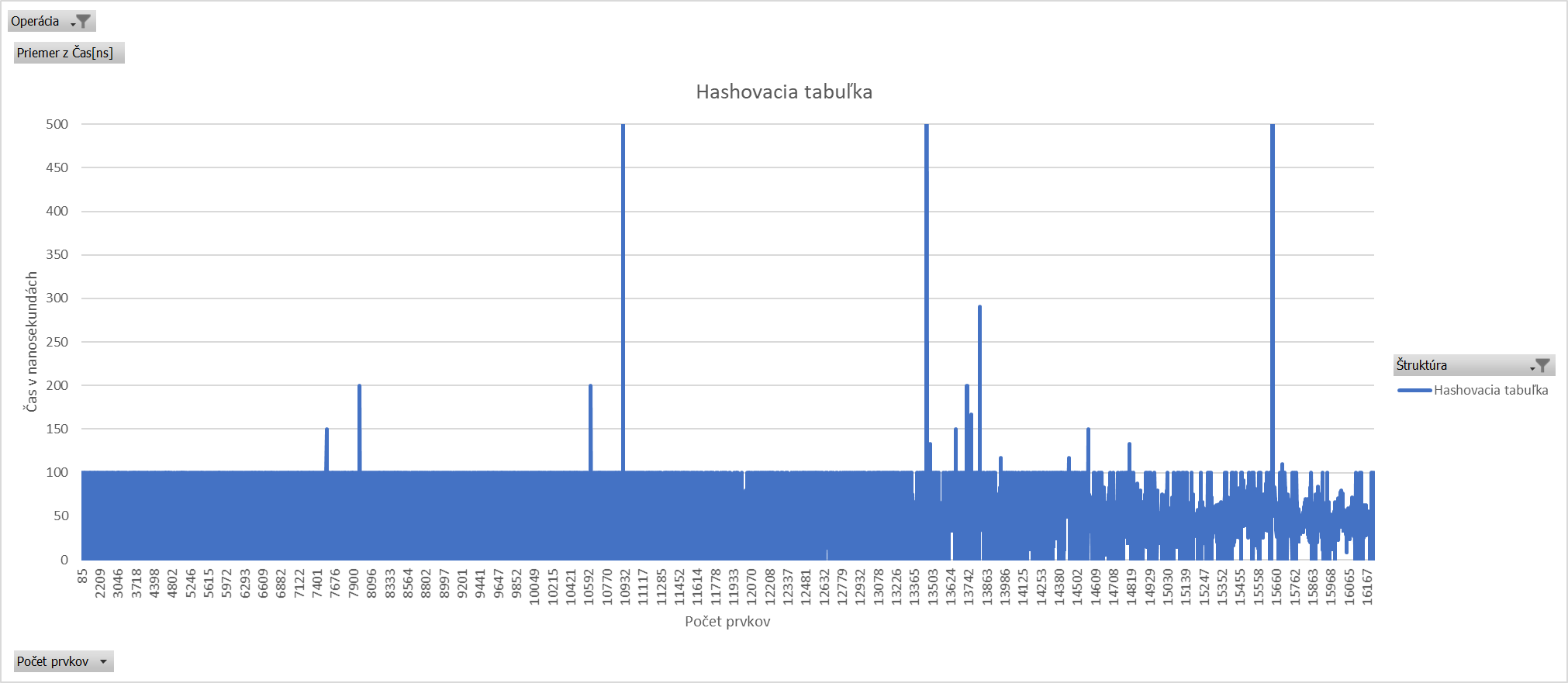


**Operácia Sprístupni prvok na základe kľúča**



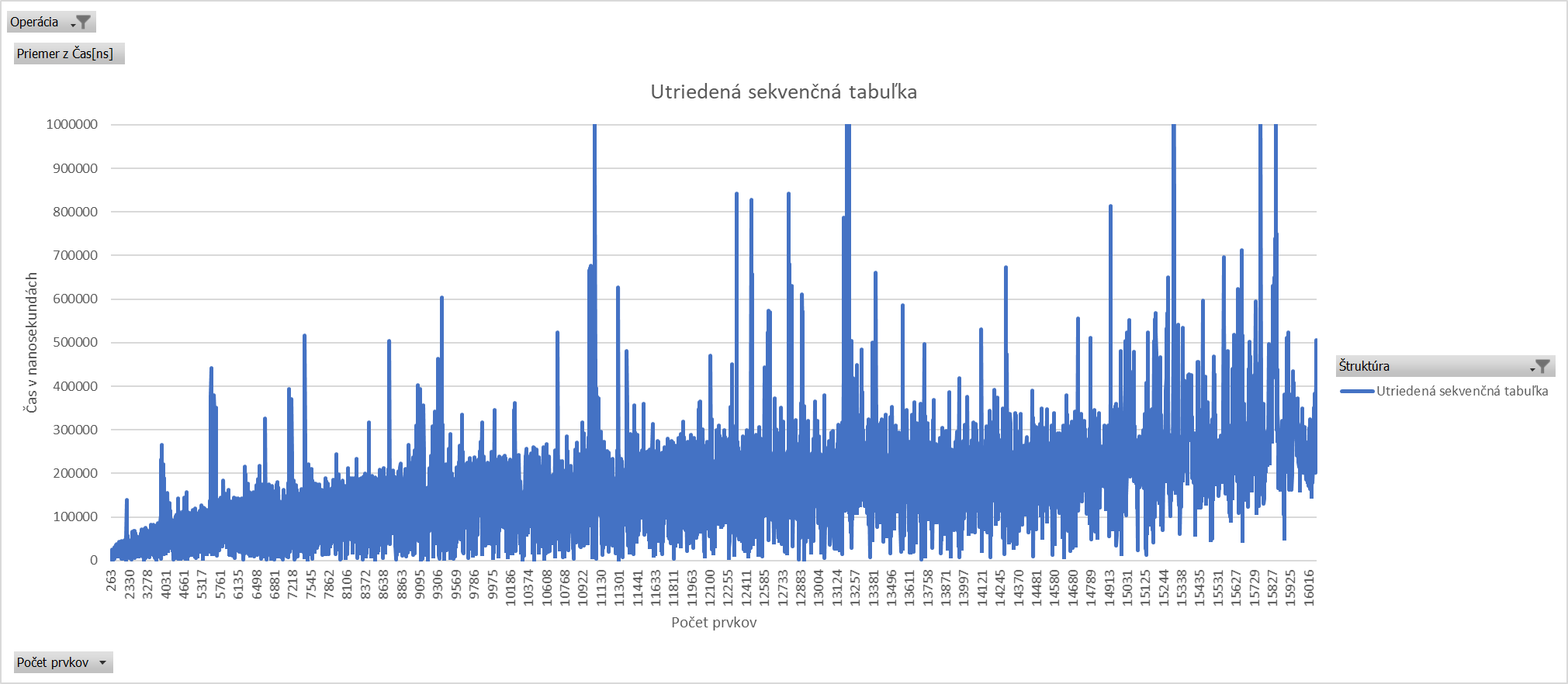


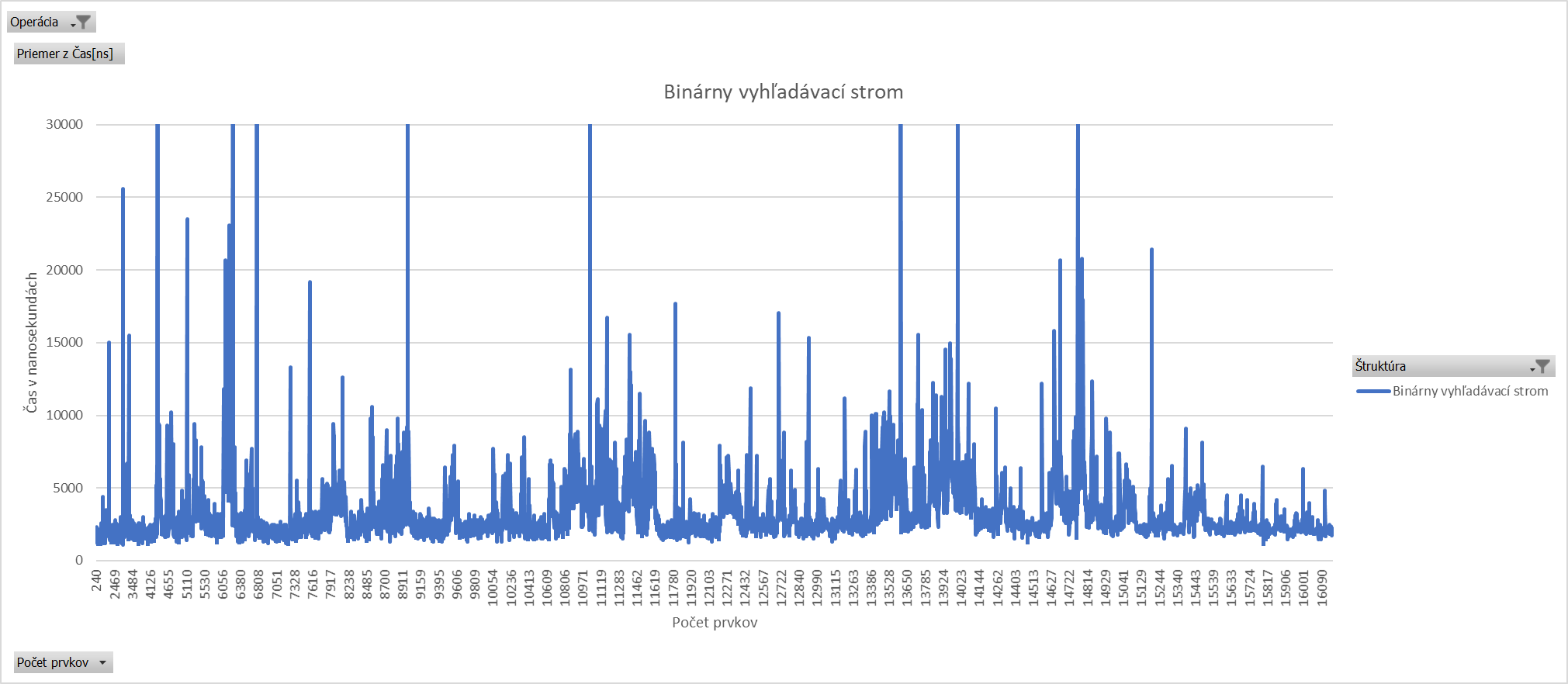


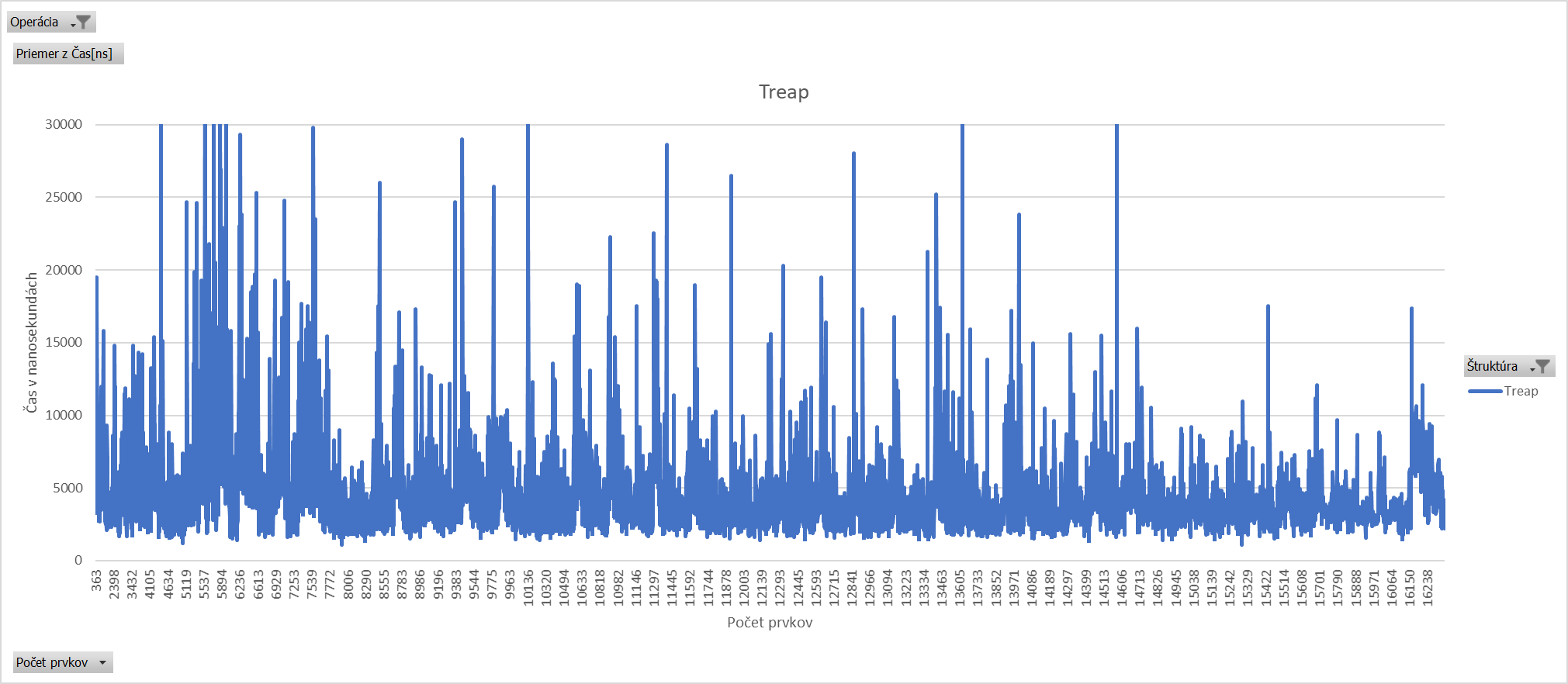


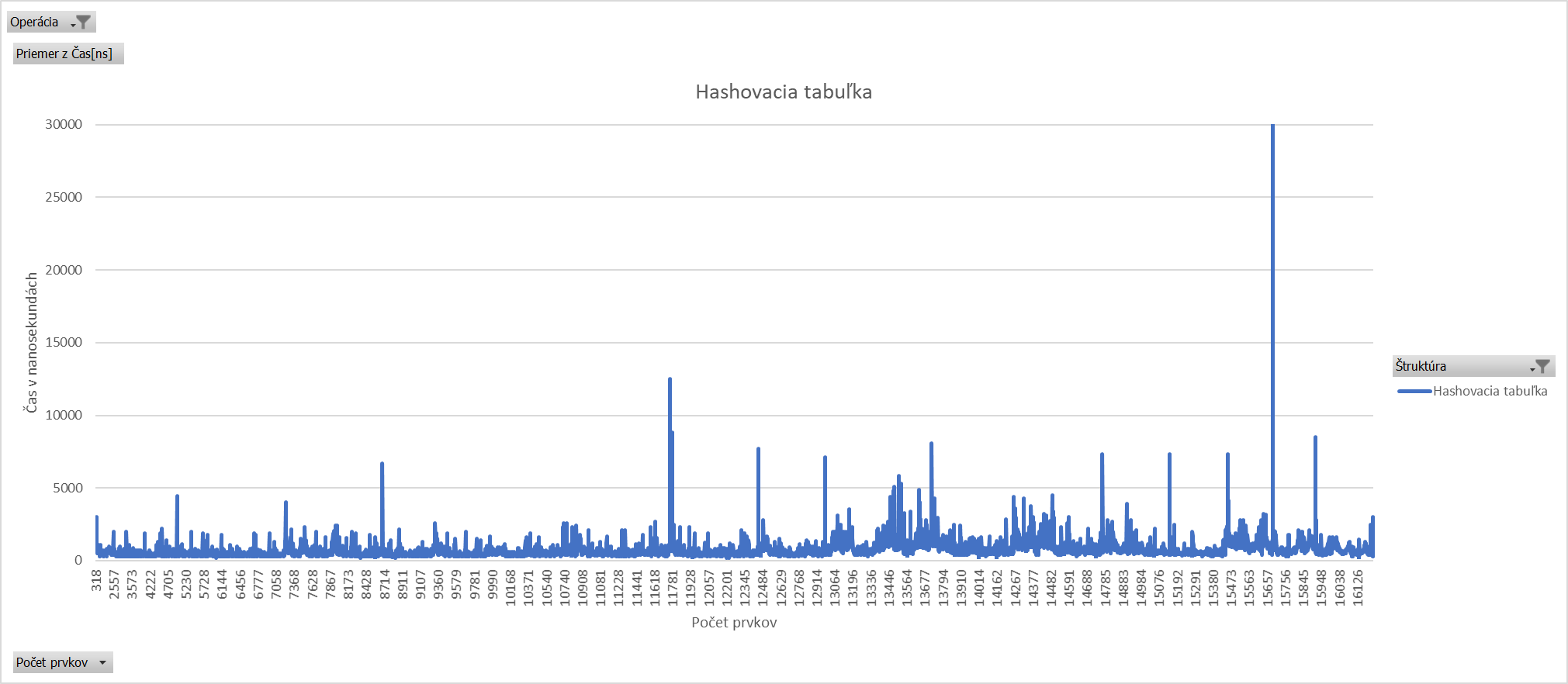
**Operácia Odstráň prvok na základe kľúča**

(POZOR: škálovanie osi y v je v tomto prípade pri utriedenej sekvenčnej tabuľke iné ako pri ostatných tabuľkách)

****

****

****

****

Nie zo všetkých grafov by sa dala odhadnúť zložitosť, ktorá by zodpovedala teoretickej zložitosti, najmä pri operácii sprístupni všetky operácie vyzerajú ako jednotkové, aj keď sú logaritmické pri všetkých tabuľkách okrem hashovacej.

## Analýza vhodnosti testovaných štruktúr v jednotlivých scenároch

Pre porovnanie výkonu štruktúr som každý scenár spustil 10-krát, odmeral časy vykonávania scenárov a z týchto časov som vypočítal priemery, ktoré možno vidieť v tabuľke a grafe nižšie. Môžeme vidieť, že hashovacia tabuľka je pri vysokom počte prvkov najrýchlejšia vo všetkých scenároch, kvôli jednotkovej zložitosti vykonávania operácií, nasleduje binárny vyhľadávací strom, ktorý je o trochu rýchlejší ako treap a nakoniec je tu utriedená sekvenčná tabuľka, ktorá je najpomalšia, kvôli Nkovej zložitosti vkladania a vyberania prvkov. Táto štruktúra je preto vhodná ak chceme mať utriedené dáta v kompaktnej pamäti a budeme k dátam hlavne pristupovať, teda nebudeme často vkladať ani vyberať prvky.

|  |  |  |  |
| --- | --- | --- | --- |
| **štruktúra** | **priemerný čas vykonávania scenárov v sekundách** | | |
| **A** | **B** | **C** |
| **Utriedená sekvenčná tabuľka** | 3,071 | 8,807 | 5,528 |
| **Binárny vyhľadávací strom** | 0,092 | 0,215 | 0,148 |
| **Treap** | 0,171 | 0,364 | 0,266 |
| **Hashovacia tabuľka** | 0,035 | 0,057 | 0,044 |