**Druhý semestrálny projekt**

**Algoritmy a údajové štruktúry 2**

Tomáš Vyšinský

5ZII11

Obsah

[Úvod 3](#_Toc153505044)

[UML Diagram 4](#_Toc153505045)

[Počty prístupov v triede DynamicHashFile 5](#_Toc153505046)

[Insert 5](#_Toc153505047)

[Find 5](#_Toc153505048)

[Delete 5](#_Toc153505049)

[Edit 5](#_Toc153505050)

# Úvod

Cieľom tohto projektu bolo vytvoriť jednoduchý softvér na správu realitných objektov, konkrétne budov a parcelov s grafickým používateľským rozhraním za pomoci samostatne naprogramovanej údajovej štruktúry.

# UML Diagram

Pre veľkosť UML Diagramu a jeho lepšiu prehľadnosť bude uložený aj vo formáte .png ako príloha.



# Počty prístupov v triede DynamicHashFile

## Insert

Na začiatku insertu sa nájde node v Trie, ktorý obsahuje adresu k prvému blocku v prípade, že nejaký alokovaný.

### Pokiaľ nemá block alokovaný

Využije metódu getRemovedFreeBlock zavolanú nad hlavným súborom:

Zoberie prázdny block. V prípade, že je zreťazenie prázdnych blockov prázdne ho tento úkon stojí dva prístupy. Jeden na zväčšenie súboru a jeden na vloženie prázdneho blocku.

V prípade, že zreťazenie voľných blockov obsahuje prázdne blocky, stojí ho tento úkon 2 alebo 4 prístupy podľa toho, či je v zreťazení jeden alebo viac blockov.

Následne potrebuje ešte 2 prístupy na zapísanie nového záznamu do blocku.

### Pokiaľ má block alokovaný

Započne cyklus hľadania správneho blocku, ktorý trvá dokým sa prvok nevloží alebo dokým sa počas cyklu nevyskytne situácia, na základe ktorej nebude možné vložiť block do súboru.

Najprv sa kontroluje, či netreba rozmnožiť node.

Ak áno, stojí ho jeden prístup načítanie pôvodného blocku.

Následne ho uvolní pomocou metódy freeTheBlock, zavolanej and hlavným súborom, čo ho stojí dva až n+1 prístupov, kde n je počet blockov v zreťazení prázdnych blockov v prípade, že nie je na konci súboru. V prípade, že sa na konci súboru nachádza ho táto operácia stojí 2 až n\*4 prístupov kde n je počet voľných blockov v zreťazení voľných blockov.

Pri priradzovaní blockov novým potomkom zavolá jeden až dva krát metódu getRemovedFreeBlock, ktorá ho stojí 2 alebo 4 prístupy za zavolanie.

Následne zapíše nové blocky nodov na čo potrebuje jeden až dva prístupy.

Pokiaľ našiel node, do ktorého treba vložiť prvok, potrebuje dva prístupy pokiaľ bude vkladať do blocku v hlavnom súbore, najviac k+1 prístupov pokiaľ existuje voľné miesto v niektorom z preplňovacích blockov a k + 4 až 6 prístupov pokiaľ alokuje nový block v zreťazení kde k je počet blockov v pôvodnom zreťazení.

## Find

Metóda find potrebuje 1 až k prístupov do súborov kde k je celkový počet zreťazených blockov.

## Delete

Na samotné zmazanie z blocku sú potrebné 2 až k+1 pristupov do suboru kde k je počet blockov v zreťazení.

Následne môže prebehnúť striasanie v prípade, že je to potrebné. V ňom najprv prebehne k – p prístupov kde p je počet už prejdených blockov v zreťazení aby sme sa dostali na koniec zreťazenia. Príslušný block sa uvolňuje pomocou metódy freeTheBlock zavolanej na preplňovacom súbore, čo stojí 2 až n\*4 prístupov kde n je počet zreťazených voľných blockov v preplňovacom súbore. Jeden prístup na zapísanie novej informácie do predchádzajúceho súboru a 2 až k\*2 prístupov pre zapísanie záznamov z vymazaného blocku.

V prípade, že je potrebný merge nodov, potrebuje 0 až h\*(3 + (2 až n\*4 za metódu freeTheBlock)) prístupov kde h je maximálna výška Trie.

## Edit

Metóda Edit potrebuje 1 až k+1 prístupov do súborov kde k je celkový počet zreťazených blockov.