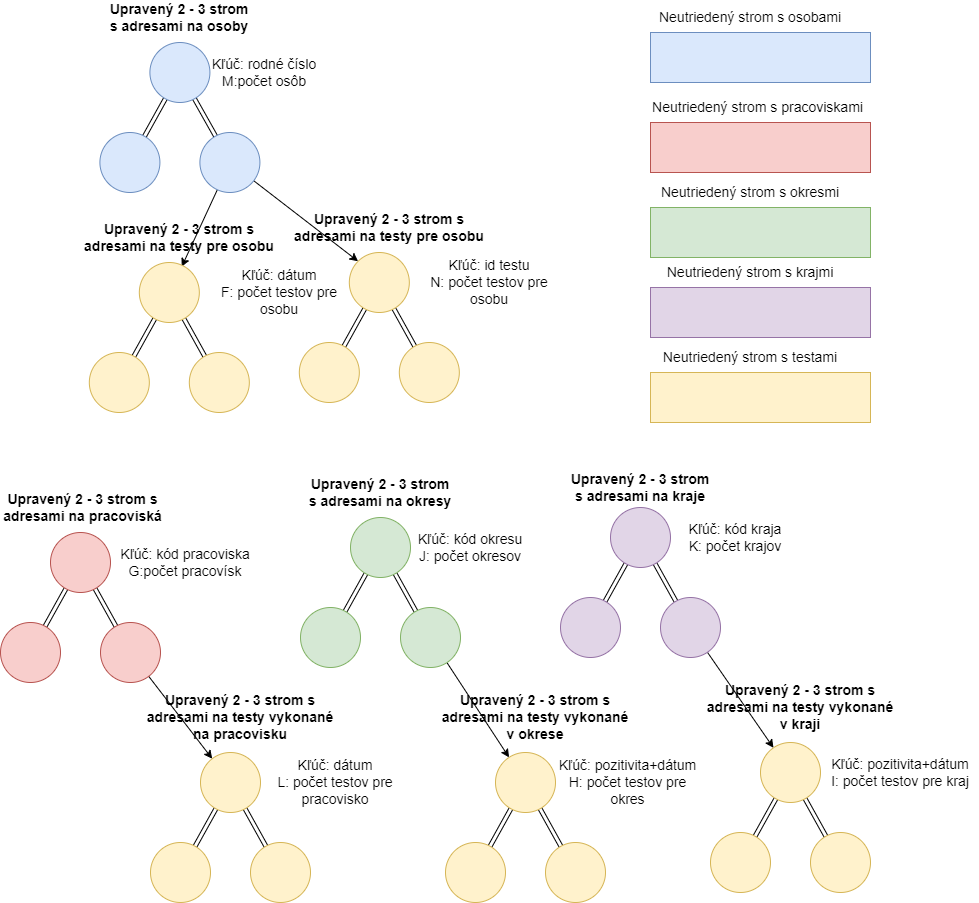
**ŽILINSKÁ UNIVERZITA V ŽILINE**

FAKULTA RIADENIA A INFORMATIKY

**Algoritmy a údajové štruktúry 2**

**Semestrálna práca 2**

Martin Kostelej 560719



Obrázok Nákres systému

**Bližší popis implementácie neutriedeného súboru a upraveného 2-3 stromu aby uchovával údaje v súbore:**

**Neutriedený súbor:** funguje na princípe, že sa na začiatku zapíše do súboru hlavička, ktorá obsahuje počet záznamov, ktoré sú aktuálne uložené v súbore(zarátava aj prázdne, teda nevalidné miesta) a adresu, kam sa má uložiť v poradí ďalší záznam. Strom si ďalej uchováva adresy, kde sú prázdne miesta. Využívajú sa dva prioritné fronty: jeden zoradený zostupne a druhý vzostupne. Vzostupný sa používa pri vkladaní a zostupný pri mazaní. Pri mazaní na konci súboru sa súbor vždy skracuje o počet voľných miest, ktoré sú pred ním.

**Upravený 2-3 strom:** do hlavičky si ukladá počet použitých blokov(nodov 2-3 stromu). Taktiež ako aj pri neutriedenom súbore počíta aj nevalidné bloky. V hlavičke sa ďalej nachádza opäť adresa na ďalší vkladaný blok a adresa na koreň stromu. Rovnakým spôsobom si uchováva nevalidné bloky v dvoch neutriedených súboroch. Mazanie funguje obdobným spôsobom ako pri neutriedenom súbore: pri mazaní na konci súboru sa súbor vždy skracuje o počet voľných blokov, ktoré sú pred ním.

Každý blok má tiež hlavičku, v ktorej si ukladá informácie ohľadom bloku: referencie z pôvodného stromu na synov a otca boli nahradené adresami týchto blokov. Ďalej sa v hlavička nachádza informácia či je blok trojvrchol, či je validný a ešte svoju adresu. Samotné dáta v bloku sú rozdelené na dve časti: v prvej je kľúč prvého údaju v bloku a jeho údaje a v druhej časti je druhý kľúč s jeho údajmi.

**Zložitosti operácií:**

**1.Vkladanie PCR testu:** O(log **m**)+O(log **n**)+O(log **n**)+O(log **j**)+O(log **h**)+O(log **k**)+O(log **i**)+O(log **g**)+O(log **l**), kde **m** je počet osôb, **n** počet testov pre osobu, **j** počet okresov, **h** počet testov pre okres, **k** počet krajov, **i** počet testov v kraji , **g** počet pracovísk a **l** počet testov pre pracovisko.

* O(log **m**): nájdenie osoby
* O(log **n**): vloženie testu pre osobu do stromu, kde je kľúčom jeho id
* O(log **n**): vloženie testu pre osobu do stromu, kde je kľúčom jeho dátum
* O(log **j**)+O(log **h**): nájdenie okresu a vloženie testu
* O(log **k**)+O(log **i**): nájdenie kraju a vloženie testu
* O(log **g**)+O(log **l**): nájdenie pracoviska a vloženie testu

**2.Vyhľadanie výsledku testu pre pacienta:** O(log **m**)+O(log **n**), kde **m** je počet osôb a **n** počet testov pre danú osobu.

**3.Výpis testov pre pacienta:** O(log **m**)+O(**n**), kde **m** je počet osôb a **n** počet testov pre danú osobu.

**4.Výpis pozitívnych testov za obdobie pre okres:** O(log **j**)+O(log **h**)+O(**z**), kde **j** je počet okresov, **h** počet testov v okrese a **z** počet hodnôt testov, ktoré spĺňanú daný dátum a sú pozitívne.

**5.Výpis testov za obdobie pre okres:** O(log **j**)+O(log **h**)+O(**z**) + O(log **j**)+O(log **h**)+O(**y**), kde **j** je počet okresov, **h** počet testov v okrese, **z** počet testov, ktoré spĺňajú daný dátum a sú pozitívne a **y** počet testov, ktoré spĺňajú dátum a sú negatívne.

**6.Výpis pozitívnych testov za obdobie pre kraj:** O(log **k**)+O(log **i**)+O(**z**), kde **k** je počet krajov, **i** počet testov pre kraj a **z** počet testov, ktoré spĺňajú daný dátum a sú pozitívne.

**7.Výpis testov za obdobie pre kraj:** O(log **k**)+O(log **i**)+O(**z**)+O(log **k**)+O(log **i**)+O(**y**), kde **k** je počet krajov, **i** počet testov pre kraj, **z** počet testov, ktoré spĺňajú daný dátum a sú pozitívne a **y** počet testov, ktoré spĺňajú dátum a sú negatívne.

**8.Výpis všetkých pozitívnych testov za obdobie:** O(**k** \* log **i** \* **z**), kde **k** je počet krajov, **i** počet testov pre kraj a **z** počet testov, ktoré spĺňajú dátum a sú pozitívne.

**9.Výpis všetkých testov za obdobie:** O(**k** \* log **i** \* **z**) + O(**k** \* log **i** \* **y**), kde **k** je počet krajov, **i** počet testov pre kraj, **z** počet testov, ktoré spĺňajú dátum a sú pozitívne **y** počet testov, ktoré spĺňajú dátum a sú negatívne.

**10.Výpis chorých v okrese k dátumu:** O(log **j**)+ )+O(log **h**)+O(**z**), kde **j** je počet okresov, **h** počet testov v okrese a **z** počet hodnôt testov, ktoré spĺňanú daný dátum a sú pozitívne.

**11.Výpis chorých v kraji k dátumu:** O(log **k**)+O(log **i**)+O(**z**), kde **k** je počet krajov, **i** počet testov pre kraj a **z** počet testov, ktoré spĺňajú daný dátum a sú pozitívne.

**12.Výpis chorých k dátumu:** O(**k** \* log **i** \* **z**), kde **k** je počet krajov, **i** počet testov pre kraj a **z** počet testov, ktoré spĺňajú dátum a sú pozitívne.

**13.Výpis okresov podľa počtu chorých k dátumu:** O(**j** \* log **h** \* **z**)+O(**j** \* log **x**)+O(**j**), kde **j** je počet okresov, **h** počet testov v okrese, **z** počet chorých pre dátum a **x** aktuálny počet prejdených okresov.

* O(**j** \* log **h** \* **z**): prejdenie všetkých okresov a nájdenie pozitívnych
* O(**j** \* log **x**): postupné pridávanie prejdených okresov aj s ich počtom do nového stromu
* O(**j**): prejdenie nového stromu s okresmi a s počtami chorých

**14.Výpis krajov podľa počtu chorých k dátumu:**

O(**k** \* log **i** \* **z**)+O(**j** \* log **x**)+O(**j**), kde **k** je počet krajov, **i** počet testov v kraji, **z** počet chorých pre dátum a **x** aktuálny počet prejdených krajov.

* O(**k** \* log **i** \* **z**): prejdenie všetkých krajov a nájdenie pozitívnych
* O(**k** \* log **x**): postupné pridávanie prejdených krajov aj s ich počtom do nového stromu
* O(**k**): prejdenie nového stromu s krajmi a s počtami chorých

**15.Výpis testov za obdobie na pracovisku:** O(log **g**)+O(log **l**)+O(**z**), kde **g** je počet pracovísk, **l** počet testov v pracovisku a **z** počet testov, ktoré vyhovujú danému obdobiu.

**16.Vyhľadanie PCR testu:** O(**m**)+O(log **n**), kde **m** je počet osôb a **n** počet testov pre osobu.

**17.Vloženie osoby:** O(log **m**), kde **m** je počet osôb.

**18.Vymazanie PCR testu:** O(**m**)+O(log **n**)+O(log **h**)+O(log **i**)+O(log **l**)+O(log **n**)+O(log **n**), kde **m** je počet osôb, **n** je počet testov pre osobu, **h** počet testov pre okres, **i** počet testov pre kraj, **l** počet testov pre pracovisko, **n** počet testov pre osobu.

* O(**m**)+O(log **n**): nájdenie testu
* O(log **h**): mazanie testu v okrese
* O(log **i**): mazanie testu v kraji
* O(log **l**): mazanie testu v pracovisku
* O(log **n**)+O(log **n**): mazanie testu pre osobu(strom kde je kľúčom id a strom kde je kľúčom dátum)

**19.Vymazanie osoby aj s testami:** O(log **m**)+O(**n**)+O(**n**\*[zložitosť s predchádzajúcej operácie])+O(log **m**), kde **m** je počet osôb, **n** počet testov pre osobu.

* O(log **m**)+O(**n**): nájdenie osoby a prejdenie všetkých testov
* O(**n**\*[zložitosť s predchádzajúcej operácie]): každý test maže zo všetkých stromov
* O(log **m**): mazanie osoby

**Počet prístupov do súboru pre 2-3 strom:**

**Insert:**

* Pokiaľ je **list** do ktorého sa vkladá **dvojvrchol**: na začiatku sa postupne hľadá list, do ktorého by vložil, čiže **log n** čítaní**,** pre n validnych blokov v súbore + 1 zápis.
* **List** do ktorého sa vkladá je **trojvrchol**: na začiatku sa postupne hľadá list, do ktorého by vložil, čiže **log n** čítaní**,** pre **n** validnych blokov v súbore. Potom postupne pristupuje od listu v súbore k parentovi a v každom musí čítať dvoch potomkov. Takže **m\*2** čítaní, kde **m** je počet parentov, kým nájde miesto. V každom prechádzaní cez parenta musí ešte upravovať údaje pre 4 rôzne bloky , t.j. **m\*4**(niekedy len 2, nedá sa presne určiť) zápisov. Výsledne teda **log n + m\*2** čítaní a **m\*4** zápisov.

**Find:** počet čítaní zo súboru je **log n** pre n počet validnych blokov v súbore.

**Delete:**

* Pokiaľ je v strome jediný prvok tak nie je potrebný žiadny zápis. Len sa nastaví dĺžka súboru na veľkosť hlavičky a upravia sa údaje v hlavičke. V mojom fungovaní sa hlavička neaktualizuje v súbore pri každej zmene ale len keď sa aplikácia vypína, aby sa redukoval počet prístupov do súboru. Vytvorená metóda endWorkWithFile, ktorá ho updatne s aktuálnymi údajmi pre hlavičku a zatvorí.
* Ak je trojvrchool a list tak **1** zápis.
* Ak nie je nenastávajú predchádzajúce 2 situácie, tak **m** čítaní, kde **m** je počet prechodov kým je v liste.
  + Keď je nájdený list trojvrchol, tak + 2 zápisy. **Výsledne**: **m** čítaní a **2** zápisy
  + Ak nie je trojvrchol tak tiež 2 zápisy a postupne bude prechádzať od tohto vrcholu cez parentov, takže **n** čítaní kde **n** je počet potrebných prechodov k parentovi. Počas prechádzania ešte hľadá aj bratov pre daný vrchol, kde počet čítaní bude v najhoršom prípade **4**. Ešte sa vykonáva **1** čítanie pre aktuálneho vrcholu. Počas prechádzania cez vrcholy nastáva situácia že budú potrebné **3** zápisy. Najhoršia možnosť prístupov do tohto momentu je teda: **m+n\*(4+1)** čítaní a **n\*3** zápisov.
    - Pokiaľ sa dostaneme až ku koreňu tak sa vykonáva **1** čítanie a **2** zápisy. Výsledne: **m+n\*(4+1)+1** čítaní a **n\*3**+**2** zápisy
    - Na konci sa ešte vykonajú **4** zápisy a **1** čítanie. **Výsledne**: **m+n\*(4+1)+1** čítaní a **n\*3**+**4** zápisy

**Interval search:**

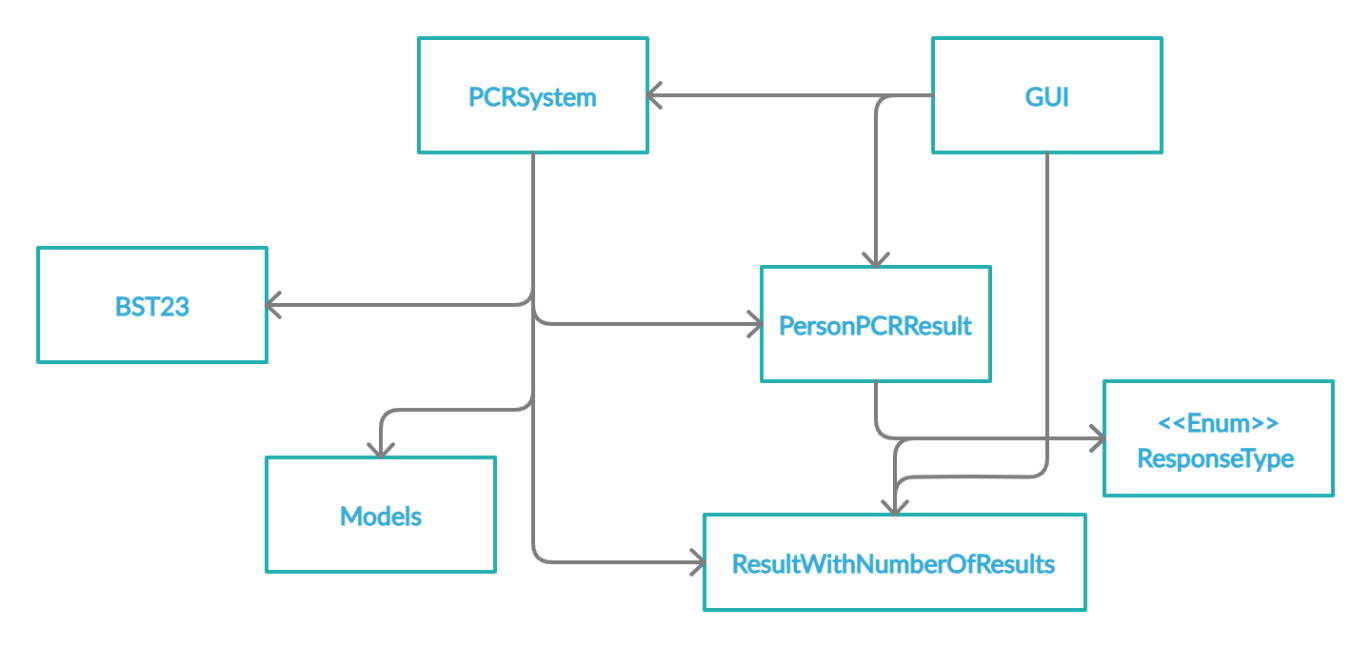
Log **m** čítaní, kde **m** je počet prejdených nodov, kým nájde ten na ktorom začne. Potom **n\*k**, kde **n** je počet prvkov patriacich do intervalu a **k** je počet prvkov potrebných na prejdenie k inOrdeer nasledovníkovi. **Výsledne: log m + n\*k**. čítaní.

**Popis mojej implementácie intervalového vyhľadávania:** funkcia na intervalové vyhľadávanie má 2 parametre: node s minimálnou hodnotou a node s maximálnou hodnotou. Na začiatku sa vytvorí array list, do ktorého sa budú vkladať nájdené nody a na konci prehľadávania tieto hodnoty vráti. Prvý krok je prechádzanie stromu tak, že sa pokúša nájsť minimálnu hodnotu. Ak sa dostal až do listu stromu a taký node nenašiel, tak si zoberie list, overí či patrí do požadovaného intervalu. Ak patrí, tak vloží hodnotu do array listu a ďalej bude pokračovať v strome na in order nasledovníka. Takto isto postupoval ak node s minimálnou hodnotou našiel.

Hľadanie in order nasledovníka funguje na princípe toho, že sa do funkcie posiela naposledy nájdený node.

* Najprv sa overuje či je daný node trojvrchol. Pokiaľ nie je, tak skontroluje či má pravého syna. Pokiaľ ho má, tak do neho vojde a postupne zlieza až k listu cez ľavých synov. Tuto nachádza in order nasledovníka. Ak syna pôvodný node nemal, tak pôjde po rodičoch a tam overuje či ich hodnoty už boli použité v in order poradí alebo sú jeho nasledovníci. Toto sa zisťuje podľa toho aká je hodnota v porovnaní s rodičovskými hodnotami(ak sa vraciame z pravej strany tak rodič už bol predtým použitý a preto lezieme vyššie).
* Pokiaľ je trojvrchol, tak sa najprv zisťuje, či je aktuálna hodnota, pre ktorú hľadáme nasledovníka na ľavej pozícii.
  + Ak je a nemá stredného syna, tak vracia pravú hodnotu nodu. Ak ale stredného syna má, tak do neho vlezie a zlieza do ľavých synov až kým je v liste. Danú hodnotu potom vráti.
  + Ak je na pravej pozícii, tak zistí či má pravého syna. Keď má, tak zlieza po ňom stále doľava až kým je v liste. Tam nachádza in order nasledovníka. Ak ale pravého syna nemá, tak pôjde po rodičoch. Kde postupuje v podstate takým istým spôsobom, ako pri podobnej situácii pri dvojvrcholoch.

Takto stále pokračuje v hľadaní nasledovníkov a pridáva ich do array listu, pokiaľ patria do intervalu. Prehľadávanie končí, keď dôjde na koniec stromu alebo daná hodnota už nepatrí do intervalu. Následne vracia array list s hodnotami ktoré medzi zadané minimum a maximum.



Obrázok 2 Nákres tried systému

Hlavná trieda systému je **PCRSystem,** ktorá má na starosti všetku hlavnú funkcionalitu. Sú v nej všetky funkcie na vyhľadávanie, pridávanie, mazanie osôb, testov, krajov... Taktiež sa tu nachádzajú všetky stromy kde sa ukladajú adresy pre osoby, testy, okresy, kraje a pracoviská. Adresy odkazujú na 5 vytvorených neutriedených súborov, kde sa ukladajú už konkrétne údaje.

**GUI** časť pozostáva z viacerých tried a formov, ktoré špecifikujú fungovanie a výzor používateľského rozhrania. Táto časť má na starosti minimálnu logiku a jej fungovanie je na základe typu odpovede od PCRSystemu.

**PersonPCRResult** je objekt, ktorý uchováva typ odpovede(**Response type**) a výstupný string, ktorý potom GUI zobrazí. **ResultWithNumberOfResults**  funguje na rovnakom princípe ale má ešte pridanú hodnotu pre počet výsledkov, ktoré daná odpoveď vráti. Toto sa využívalo keď sa komplikovalo zobrazenie počtu výsledkov, keď sa brali z viacerých stromov.

Časť systému **models** je tvorená triedami pre osoby, kraje, okresy, pracoviská a testy, ktoré sa v triede **PCRSystem** ukladajú do stromov. V každej z nich sa uchovávajú potrebné hodnoty aby sa dalo jednoducho pristupovať k údajom(napr. meno, priezvisko, dátum narodenia, dátum testu, referencia na test, poprípade na okres, kraj alebo pracovisko...). Ďalej sa v tejto časti nachádzajú všetky kľúče pre stromy. Ďalej sú tu aj dátove triedy, ktoré obaľujú vždy hodnoty ako napríklad osobu a jej kľúč. Tieto obalené dáta sa potom vkladajú do stromov. Obsahuje aj triedu adresa, ktorá sa ukladá do stromov(obsahuje adresu na potrebný údaj v neutriedenom súbore).

**BST23** je implementovaný 2-3 strom upravený na uchovávanie údajov v súboroch, ktorý sa využíva na ukladanie všetkých adries.