# Použitie údajových štruktúr (úroveň 3):

Na uloženie územných jednotiek som medzi nimi vytvoril stromovú hierarchiu. Najvyššia územná jednotka – štát, je koreňom. Každá územná jednotka má atribút SortedSequenceTable children\_, čo sú nižšie územné jednotky, ktoré k nej prislúchajú.

Vybral som SortedSequenceTable, kvôli prístupu k prvkom, ktorý má zložitosť O(logN) – pretože v tabuľke sa budú najmä hľadať údaje, nie modifikovať. V tabuľke bude kľúčom názov územnej jednotky.

# UML (dáta + UI):

Diagram

Description automatically generated

UML (filtre + kritériá):

Diagram

Description automatically generated

**Trieda TerritorialUnit**

Predstavuje územnú jednotku, v ktorej budú uložené všetky potrebné dáta. Jednotlivé územné jednotky sú jej potomkami. Údaje o počtoch ľudí s daným vzdelaním su uložené v atribúte Array<int>\* educationCount. V Arrayi budú počty ľudí s daným vzdelaním v nasledujúcom poradí: *osoby vo veku 0-14 rokov (abs.);základné vzdelanie (abs.);stredné odborné (učňovské) vzdelanie (bez maturity) (abs.);úplné stredné vzdelanie (s maturitou) (abs.);vyššie odborné vzdelanie (abs.);vysokoškolské vzdelanie (abs.);bez školského vzdelania – osoby vo veku 15 rokov a viac (abs.);nezistené (abs.) -* Array bude mať teda vždy 8 stĺpcov.

V gettri getEducationCount, ako parameter bude vstupovať hodnota int type v rozmedzí <0, 7>. Podľa tejto hodnoty sa vráti stĺpec v Arrayi educationCount\_, kde sa bude nachádzať prislušný počet ľudí s daným vzdelaním.

**Trieda Data**

Bude obsahovať (atribút) štát - koreň stromovej hierarchie územných jednotiek. Má operáciu loadDataFromFile(), ktorá načíta všetky údaje o územných jednotkách prislúchajúce štátu zo súboru.

**Trieda Sorter**

Má operáciu sort(UnsortedTable, criterion, bool ascending) - utriedi UnsortedTable s územnými jednotkami podľa zadaného kritéria.

**Trieda CompositeFilter**

Vytvorí kompozitný filter, ktorý uchováva vložené filtre. Má operácie :

bool passFilter\_AND(ObjectType& o)

bool passFilter\_OR(ObjectType& o)

Ktoré determinujú či objekt spĺňa filtre (AND – všetky, OR – aspoň jeden).

**Trieda UserInteraction**

Má na starosti interakciu s uživateľom. V CLI Ponúkne akú funkcionalitu (Triedenie/Filtrovanie) má program vykonať a s pomocou triedy **FilterMakerUI** vytvorí uživateľom špecifikované filtre.

# **Popis výpočtu každého triediaceho kritéria a ich časové zložitosti.**

I.𝑲𝑵á𝒛𝒐𝒗

Ω⊆𝑂𝐵𝐶𝐸∪𝑂𝐾𝑅𝐸𝑆𝑌∪𝐾𝑅𝐴𝐽𝐸

Π=∅

𝜅=𝜔.𝑛á𝑧𝑜𝑣;

τ=𝑠𝑡𝑟𝑖𝑛𝑔

v tomto kritériu nájdem v stromovej hierarchii územnú jednotku (Npr. pri hľadaní obce prejdem všetky kraje, v krajoch prejdem okresy a v okresoch hľadám obec), a vrátim stringom jej názov. Nájdenie obce v utriedenej tabuľke má zložitost O(PocetKrajov + PocetOkresov + logN). Nájdenie okresu O(PocetKrajov + logN). Nájdenie Kraju je O(logN).

**ZLOŽITOSŤ KRITÉRIA:**

Vypísanie ‘k’ položiek je O(1) \* k. Časová zložitosť kritéria je O(1) \* k.

IV.𝑲𝑼𝑱𝑽𝒛𝒅𝒆𝒍𝒂𝒏𝒊𝒆𝑷𝒐č𝒆𝒕

Ω⊆𝑂𝐵𝐶𝐸∪𝑂𝐾𝑅𝐸𝑆𝑌∪𝐾𝑅𝐴𝐽𝐸

Π=(𝑣𝑧𝑑𝑒𝑙𝑎𝑛𝑖𝑒∈𝑉𝑍𝐷𝐸𝐿𝐴𝑁𝐼𝐸)

𝜅=𝜔.𝑝𝑜č𝑒𝑡𝑂𝑏𝑦𝑣𝑎𝑡𝑒ľ𝑜𝑣𝑆𝑜𝑉𝑧𝑑𝑒𝑙𝑎𝑛í𝑚(𝑣𝑧𝑑𝑒𝑙𝑎𝑛𝑖𝑒)

τ=𝑖𝑛𝑡𝑒𝑔𝑒𝑟

podľa parametra typ vzdelania, vrátim počet ľudí na územnej jednotke s daným typom vzdelania. Nájdenie obce v utriedenej tabuľke v stromovej hierarchii má zložitost O(PocetKrajov + PocetOkresov + logN). Nájdenie okresu O(PocetKrajov + logN). Nájdenie Kraju je O(logN).

**ZLOŽITOSŤ KRITÉRIA:**

Vypísanie počtu ľudí s daným vzdelaním (prístup k prvku v Arrayi) je O(1). Zložitosť kritéria je teda O(1).

V.𝑲𝑼𝑱𝑽𝒛𝒅𝒆𝒍𝒂𝒏𝒊𝒆𝑷𝒐𝒅𝒊𝒆𝒍

Ω⊆𝑂𝐵𝐶𝐸∪𝑂𝐾𝑅𝐸𝑆𝑌∪𝐾𝑅𝐴𝐽𝐸

Π=(𝑣𝑧𝑑𝑒𝑙𝑎𝑛𝑖𝑒∈𝑉𝑍𝐷𝐸𝐿𝐴𝑁𝐼𝐸)

𝜅=100𝐾Π𝑈𝐽𝑉𝑧𝑑𝑒𝑙𝑎𝑛𝑖𝑒𝑃𝑜č𝑒𝑡(𝜔)𝜔.𝑝𝑜č𝑒𝑡𝑂𝑏𝑦𝑣𝑎𝑡𝑒ľ𝑜𝑣

τ=𝑑𝑜𝑢𝑏𝑙𝑒∈⟨0|100⟩

podľa parametra typ vzdelania, vrátim podiel:

počet ľudí s daným vzdelaním / počet obyvateľov \* 100.

Nájdenie obce v utriedenej tabuľke v stromovej hierarchii má zložitosť O(PocetKrajov + PocetOkresov + logN). Nájdenie okresu O(PocetKrajov + logN). Nájdenie Kraju je O(logN). Vypočítanie priemeru je O(1). Vypísanie podielu ľudí s daným vzdelaním je O(1).

**ZLOŽITOSŤ KRITÉRIA:**

Vypísanie podielu ľudí s daným vzdelaním (prístup k prvku v Arrayi) je O(1). A aj výpočet podielu je O(1). Zložitosť kritéria je teda O(1).

II.𝑲𝑼𝑱𝑻𝒚𝒑

Ω⊆𝑂𝐵𝐶𝐸∪𝑂𝐾𝑅𝐸𝑆𝑌∪𝐾𝑅𝐴𝐽𝐸

Π=∅

𝜅=𝜔.𝑡𝑦𝑝Ú𝑧𝑒𝑚𝑛𝑒𝑗𝐽𝑒𝑑𝑛𝑜𝑡𝑘𝑦;

τ=𝑇𝑌𝑃\_Ú𝑍𝐸𝑀𝑁𝐸𝐽\_𝐽𝐸𝐷𝑁𝑂𝑇𝐾𝑌

Nájdem územnú jednotku a vrátim jej typ (enum TerritorialUnitType), ktorý má uložený ako atribút. Nájdenie obce v utriedenej tabuľke v stromovej hierarchii má zložitosť O(PocetKrajov + PocetOkresov + logN). Nájdenie okresu O(PocetKrajov + logN). Nájdenie Kraju je O(logN). Vrátenie typu gettrom je O(1).

**ZLOŽITOSŤ KRITÉRIA:**

Vrátenie typu gettrom je O(1).

III.𝑲𝑼𝑱𝑷𝒓í𝒔𝒍𝒖š𝒏𝒐𝒔ť

Ω⊆𝑂𝐵𝐶𝐸∪𝑂𝐾𝑅𝐸𝑆𝑌∪𝐾𝑅𝐴𝐽𝐸

Π=(𝑣𝑦šší𝐶𝑒𝑙𝑜𝑘∈𝑂𝐾𝑅𝐸𝑆𝑌∪𝐾𝑅𝐴𝐽𝐸∪𝑆𝐿𝑂𝑉𝐸𝑁𝑆𝐾𝑂)

𝜅=𝜔.𝑝𝑎𝑡𝑟í𝐷𝑜𝑉𝑦šš𝑖𝑒ℎ𝑜𝐶𝑒𝑙𝑘𝑢(𝑣𝑦šší𝐶𝑒𝑙𝑜𝑘);

τ=𝑏𝑜𝑜𝑙𝑒𝑎𝑛

Nájdem územnú jednotku a zistím, či patrí do vyššieho celku. Nájdenie obce v utriedenej tabuľke v stromovej hierarchii má zložitosť O(PocetKrajov + PocetOkresov + logN). Nájdenie okresu O(PocetKrajov + logN). Nájdenie Kraju je O(logN).

V prípade, že nájdená UJ je okres: porovnám vyšší celok územnej jednotky (atribút TerritorialUnit\* parent\_) a porovnám ho so zadaným parametrom 𝑣𝑦šší𝐶𝑒𝑙𝑜𝑘.

V prípade obce porovnám vyšší celok (parent\_ -> okres) a vyšší celok vyššieho celku územnej jednotky (parent\_->getParent\_() -> kraj) a porovnám ho so zadaným parametrom 𝑣𝑦šší𝐶𝑒𝑙𝑜𝑘.

V prípade že parameter 𝑣𝑦šší𝐶𝑒𝑙𝑜𝑘 je štát a 𝜔 nieje, tak sa vráti true.

**ZLOŽITOSŤ KRITÉRIA:**

Zistenie či UJ patrí do vyššieho celku sú pri najhoršom tri porovnania (zistenie typu parametra, porovnanie s parent\_, porovnanie s parent\_->getParent()) Zložitosť je teda 3\*O(1).

# Časová zložitosť načitávania dát:

Najprv načítam kraje O(pocetKrajov). Potom načítam okresy: pre každý okres  prechádzam kraje – hľadám zhodu v kódoch – čo je zložitosť O(pocetOkresov\*pocetKrajov). Nakoniec načítam obce: pre každú obec prechádzam okresy – hľadám zhodu v kódoch – čo je zložitosť O(pocetObci\*pocetOkresov).

Celkovo:

O(pocetKrajov + pocetOkresov\*pocetKrajov + pocetObci\*pocetOkresov)