# UNIVERZITA KONŠTANTÍNA FILOZOFA V NITRE

FAKULTA PRÍRODNÝCH VIED A INFORMATIKY

**Vizualizácia rôznych typov grafov a algoritmov z teórie grafov**

BAKALÁRSKA práca

**2022 Lukáš Hajda**

UNIVERZITA KONŠTANTÍNA FILOZOFA V NITRE

FAKULTA PRÍRODNÝCH VIED A INFORMATIKY

# Vizualizácia rôznych typov grafov a algoritmov z teórie grafov

BAKALÁRSKA práca

Študijný odbor: 18. Informatika

Študijný program: Aplikovaná informatika

Školiace pracovisko: Katedra informatiky

Školiteľ: RNDr. Júlia Tomanová, PhD.

Nitra 2022 Lukáš Hajda

# Zadanie ZP

Zadanie ZP je dokument, ktorým vysoká škola stanoví študentovi študijné povinnosti v súvislosti s vypracovaním ZP práce.

Zadanie spravidla obsahuje:

* typ záverečnej práce (ZP, DP, Rigorózna práca, Habilitačná práca a ďalšie),
* názov ZP práce,
* meno, priezvisko a tituly študenta,
* meno, priezvisko a tituly školiteľa,
* v prípade externého školiteľa meno, priezvisko a tituly konzultanta,
* školiace pracovisko,
* meno, priezvisko a tituly vedúceho pracoviska,
* anotáciu ZP práce,
* jazyk, v ktorom sa práca vypracuje,
* dátum schválenia zadania.

POVINNÉ z AIS VLOŽIŤ TU DO ZP

# Poďakovanie

Na tomto mieste môže byť vyjadrenie poďakovania napr. školiteľovi resp. konzultantovi za pripomienky a odbornú pomoc pri vypracovaní práce. Nie je zvykom ďakovať za rutinnú kontrolu, menšiu spoluprácu, alebo všeobecné rady.

NEPOVINNÉ

# ABSTRAKT

Hajda, Lukáš: Vizualizácia rôznych typov grafov a algoritmov z teórie grafov. [Bakalárska/Diplomová práca]. Univerzita Konštantína Filozofa v Nitre. Fakulta prírodných vied a informatiky. Školiteľ: RNDr. Júlia Tomanová, PhD. Stupeň odbornej kvalifikácie: Bakalár odboru Aplikovaná informatika. Nitra: FPVaI, 2022. .... s. (uviesť počet strán ZP).

Abstrakt musí byť pre čitateľa zrozumiteľný, aj bez použitia dokumentu. Je potrebné zachovať základné informácie a prístup k pôvodnému dokumentu. Abstrakt musí byť písaný ako **jeden odsek.** Treba byť čo najstručnejší a pritom dodržiavať požiadavky na obsah, ale vyhnúť sa nejasnostiam. Okrajové informácie treba používať len výnimočne, najlepšie vôbec nie. Nie je vhodné uvádzať informácie alebo tvrdenia, ktoré sa nevyskytujú v samotnom dokumente. Vždy, keď je to možné, používajú sa slovesá v činnom rode, teda v prítomnom čase. Prispieva to k jasnému, stručnému a pôsobivému písaniu. Trpný rod možno použiť na indikatívne konštatovania a na informatívne konštatovania, ak treba súčasne zdôrazniť osobu, na ktorú sa činnosť zameriava. Používa sa tretia osoba jednotného alebo množného čísla. *Terminológia:* Používajú sa významové slová z textu, ktoré pomôžu aj pri automatickom vyhľadávaní v textoch. Treba sa vyhnúť neznámym termínom, skratkám, akronymom alebo symbolom alebo ich treba definovať hneď, keď sa prvý raz vyskytnú v abstrakte. Používajú sa jednotky, symboly, terminológia ISO vždy, keď je to možné. Ak nie sú k dispozícii, používajú sa národné normy.

Kľúčové slová: (napr. Informatika. Prírodné vedy.) Minimálne 3 – 5 výrazov. Jednotlivé slová sa oddeľujú bodkou a začínajú sa veľkým písmenom)

POVINNÉ

# ABSTRACT

PRIEZVISKO, Meno autora: Názov ZP. [Bachelor/Master Thesis]. Constantine the Philosopher University in Nitra. Faculty of Natural Sciences and Informatics. Supervisor: Titul. Meno Priezvisko, Titul. Degree of Qualification: Bachelor of Applied Informatics. Nitra: FNSaI, 202x. .... p.

Text abstraktu v anglickom jazyku Text abstraktu v anglickom jazyku Text abstraktu v anglickom jazyku Text abstraktu v anglickom jazyku Text abstraktu v anglickom jazyku Text abstraktu v anglickom jazyku Text abstraktu v anglickom (100 - 150 slov) Abstrakt musí byť písaný ako jeden odsek. Text abstraktu v anglickom jazyku Text abstraktu v anglickom jazyku Text abstraktu v anglickom jazyku Text abstraktu v anglickom jazyku Text abstraktu v anglickom jazyku Text abstraktu v anglickom jazyku Text abstraktu v anglickom (100 - 150 slov) Abstrakt musí byť písaný ako jeden odsek. Text abstraktu v anglickom jazyku Text abstraktu v anglickom jazyku Text abstraktu v anglickom jazyku Text abstraktu v anglickom jazyku Text abstraktu v anglickom jazyku Text abstraktu v anglickom jazyku Text abstraktu v anglickom (100 - 150 slov).

Keywords: Informatics. Key. MS Word.

POVINNÉ

# Obsah (dopracujte až na záver ZP)

[Vizualizácia rôznych typov grafov a algoritmov z teórie grafov 2](#_Toc99311122)

[Zadanie ZP 3](#_Toc99311123)

[Poďakovanie 4](#_Toc99311124)

[ABSTRAKT 5](#_Toc99311125)

[ABSTRACT 6](#_Toc99311126)

[Obsah (dopracujte až na záver ZP) 7](#_Toc99311127)

[zoznam ilustrácií a tabuliek 9](#_Toc99311128)

[zoznam skratiek a značiek 10](#_Toc99311129)

[slovník pojmov 11](#_Toc99311130)

[Úvod 12](#_Toc99311131)

[1 Analýza súčasného stavu (pracovný názov) 13](#_Toc99311132)

[Teória grafov 14](#_Toc99311133)

[Grafová súvislosť a komponent grafu 16](#_Toc99311134)

[Najkratšia cesta 18](#_Toc99311135)

[strom, les a kostra grafu 19](#_Toc99311136)

[Algoritmy 21](#_Toc99311137)

[Prehľadávanie do ŠÍRKY 21](#_Toc99311138)

[Prehľadávanie do hĺbky 23](#_Toc99311139)

[1.1 Okraje dokumentu 24](#_Toc99311140)

[1.2 Odsadenie odsekov 24](#_Toc99311141)

[1.3 číslovanie a ODRÁŽKY 24](#_Toc99311142)

[1.4 typografické pravidlá - Dvojbodka, čiarky, bodka v texte za odrážkou 24](#_Toc99311143)

[1.5 uvádzanie citácií v texte 24](#_Toc99311144)

[1.6 Ukážka zápisu citácie v texte 25](#_Toc99311145)

[1.7 Ukážka zápisu autorov v bibliografickom zozname 25](#_Toc99311146)

[KVETINKOVÁ, V. 2016. *Teória vyučovania informatiky***.** [online] Tlačová správa, 2016. [cit. 2017-01-09]. Dostupné na internete <http://www2.statpedu.sk/buxus/ generatepage.php\_page\_id=649.htm l#pedagogi ckyproces\_kvetinkova>. (Zápis z internetovej stránky) 25](#_Toc99311147)

[2 CIELE záverečnej práce 27](#_Toc99311148)

[Podciele alebo čiastkové ciele: 27](#_Toc99311149)

[3 Metódy / postupy riešenia ZPkeho projektu 28](#_Toc99311150)

[3.1 Táto časť ZP má obsahovať: 28](#_Toc99311151)

[3.1.1 Pri opise postupu riešenia treba zachytiť: 28](#_Toc99311152)

[3.1.2 Obrázky, tabuľky a grafy 28](#_Toc99311153)

[3.2 Zápis zdrojov tabuliek, grafov, obrázkov v texte 29](#_Toc99311154)

[3.2.1 Rovnice v ZP 30](#_Toc99311155)

[4 Výsledky riešenia a ich zhodnotenie 31](#_Toc99311156)

[4.1 Časť Výsledky riešenia 31](#_Toc99311157)

[4.1.1 Časť Výsledky riešenia sa píše v minulom čase 31](#_Toc99311158)

[Záver 32](#_Toc99311159)

[Zoznam bibliografických odkazov (ukážka, ale všetky uvedené zdroje musia mať odkaz na autora v texte). 33](#_Toc99311160)

[Zoznam príloh (ukážka - strany sa nepočítajú sa do ZP) 35](#_Toc99311161)

POVINNÉ

# zoznam ilustrácií a tabuliek

Obrázok 1 22

Tabuľka 1 27

Graf 1 33

(Pridávame, ak je zoznam rozsiahly, t.j. 10 a viac.)

NEPOVINNÉ

# Úvod

V úvode autor záverečnej práce (ďalej ZP) stručne a výstižne charakterizuje stav poznania alebo praxe v oblasti, ktorá je predmetom ZP a oboznamuje čitateľa s významom ZP. Autor v úvode zdôrazňuje, prečo je práca dôležitá a prečo sa rozhodol spracovať danú tému. Autor uvedie ciele a zámery práce. V úvode autor stručne a jasne:

* charakterizuje súčasný stav poznania problematiky, ktorá je predmetom ZP,
* zdôvodní aktuálnosť témy,
* nastolí problémy, ktoré chce riešiť,
* uvedie v krátkosti ciele, ktoré sa majú v práci dosiahnuť,
* opíše použité metódy a postup riešenia,
* spresní, ktoré hlavné informačné zdroje použil,
* zdôvodní význam riešenia problematiky,
* načrtne stručný obsah jednotlivých kapitol.

Úvod predstavuje cca **1 stranu.**

Číslovanie strán začína od Úvodu.

# Analýza súčasného stavu (pracovný názov)

Analýza **súčasného stavu** riešeného problému v zahraničí a na Slovensku, je etapou riešenia v každej záverečnej práci. Analýzu problému musí obsahovať každá práca a každý študent sa musí analýzou problému pri riešení projektu zaoberať. Analýza predstavuje **30 %** ZP práce.

V ZP sa *Analýza problému* nachádza zvyčajne hneď za *Úvodom* záverečnej práce. Predstavuje teoretickú časť ZP a zahŕňa rozbor - analýzu súčasného stavu v príslušnej oblasti. Spravidla predstavuje analyticko-komparatívnu časť práce. Má byť tvorená na základe preštudovaných odborných zdrojov, nielen poskladaná z povrchne zachytených informácií z prednášok, prípadne zo skrípt alebo nejakých katalógov.

Analýzu môže byť členená do niekoľkých podkapitol, podľa toho aká problematika sa skúma. V tejto časti záverečnej práce preukazujete ako autor ZP, schopnosť používať odbornú terminológiu a preukazuje vedomosti z príslušnej oblasti. Základom je práca s bibliografickými zdrojmi, ktoré sa viažu k danej téme. Teoretická časť by mala byť (Bieliková, 2000):

* originálna,
* zachytávať  rôzne názory a prístupy relevantných autorov k uvedenej problematike,
* obsahovať komparáciu rôznych prístupov a ich zhodnotenie,
* preukazovať schopnosť samostatnej vedeckej práce bakalára,
* poskytovať obraz o súčasnom stave danej problematiky, najmä ako sa na dnešnej úrovni poznania riešia podobné problémy doma a vo svete,
* poukázať na to, aké sú s tým spojené výhody a možnosti,
* poukázať na to, aké sú s tým spojené ťažkosti a riziká,
* porovnať podobné riešenia, kategorizovať ich spolu s uvedením charakteristických atribútov a pod., podľa povahy úlohy riešenej v rámci ZP.

# Teória grafov

Veľké množstvo matematických, informatických a taktiež praktických problémov z reálneho sveta dokážeme vyjadriť a popísať pomocou abstraktného objektu pozostávajúci z dvoch časti:

* (konečná) množina bodov (vrcholov)
* množina vzájomných vzťahov (hrán)

Vrcholy môžu reprezentovať napríklad mestá na mape a hrany zas cesty medzi nimi. Ďalším príkladom môžu vrcholy značiť osobu na sociálnych sieťach a pomocou hrán vieme zistiť kto s kým je “priateľ” a kto zas nie je. [Matoušek, Nešetril, 2007]. Tento spôsob reprezentácie ponima matematický objekt nazývaný **graf**. Prvý krát bol použitý významným matematikom Leonhardom Eulerom na riešenie problému siedmych mostov v roku 1736 avšak **Teória grafov** ako samostatná matematickádisciplína vznikla až v prvej polovici dvadsiateho storočia [Paluch, 2008].

**Neorientovaný graf** :

Neorientovaný graf pozostáva z množiny vrcholov , množiny a funkcie z množiny do množiny neusporiadaných dvojic , kde .[Tomanová, ROK]

**Sled**:

Sled v grafe je ľubovoľná striedavá postupnosť hrán a vrcholov v tvare: .[Palúch,2008]

**Polosled**:

Je taky “sled”, u ktorého neberieme do úvahy orientaciu hrany. [Juránek, 2009]

**Trať**:

Postupnosť hrán nazývame **traťou** ak neprechádza dvakrát tou istou hranou. [Tomanová, ROK]

**Spojenie**:

Spojenie neorientovaných grafov , a

je graf taký, že , [Madaras, 2011]

**Cyklus**:

Je uzavretý ťah v ktorom sa okrem prvého a posledného vrchola žiaden vrchol nevyskytuje viac než raz.[Paluch, 2008]

**Slučka**:

Cyklus dĺžky 1. [Tomanová, ROK]

**Podgraf**:

Graf je podgraf grafu práve vtedy, keď , a teda .[Paluch, 2008].

**Faktor grafu**:

Graf je faktor grafu práve vtedy keď tzv. graf

obsahuje všetky vrcholy grafu . [Nečas, 1978].

Obrázok, na ktorom je text, hodinky

Automaticky generovaný popis

Obrázok 1 Neorientovaný graf

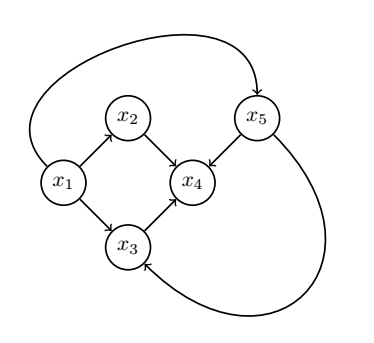
**Orientovaný graf**:

Orientovaný graf G je usporiadaná dvojica kde . [Hliněný, 2010].

**Úplny graf**:

Neorientovaný graf sa nazýva úplny, ak množina obsahuje všetky možné dvojice typu , kde a . Pri orientovaných grafoch definujeme úplny graf ak množina obsahuje všetky možné dvojice typu , kde a

.[Palúch, 2008]



Obrázok 2 Orientovaný graf

# Grafová súvislosť a komponent grafu

Ak mame graf, ktorý modeluje nejaké spojenie, tak nás zaujíma či sa vôbec dokážeme dostať z nejakého miesta na iné miesto. Ak si predstavíme dopravnú, počítačovú alebo potrubnú sieť tak je pochopiteľné , že v takýchto sieťach chceme mať možnosť sa dostať z každého miesta do iného. Grafom s touto vlastnosťou nazývame aj ako **súvislé**. Pri orientovaných grafoch ďalej definujeme pojmy ako **silne súvislý** a **slabo súvislý** graf. [Hliněný, 2010] Ďalším pojmom v teórií grafov je **komponent** grafu, vďaka ktorému dokážeme lepšie pochopiť štruktúru grafu a rozdeliť jednotlivé vrcholy do rôznych skupín a následne analyzovať tieto skupiny. Komponent grafu má rôzne využitie v reálnom živote ako napr. detekciu podvodov známy aj ako **fraud detection** [Needham, Hodler, 2019].

**Súvislosť**:

Neorientovaný graf nazývame súvislý ak medzi každou dvojicou vrcholov existuje cesta. Inak povedáne, z každého vrcholu sa dostaneme do ľubovolného iného vrcholu po nejakej ceste. [Černý, 2010]

**Silne súvisly graf:**

Orientovany graf nazývame **silno súvislý** ak pre každú dvojicu vrcholov existuje aj sled aj orientovaný sled. [Palúch, 2008] Nečas (1978) definoval orientovaný graf ako silno súvislý práve vtedy keď pre ľubovolné jeho vrcholy je kde je orientovaná vzdialenosť z vrcholu do vrcholu .

**Slabo súvislý graf**:

Orientovaný graf je **neorientovane súvislý,** alebo **slabo súvislý** ak pre každú dvojicu vrcholov existuje v polosled. [Palúch, 2020]

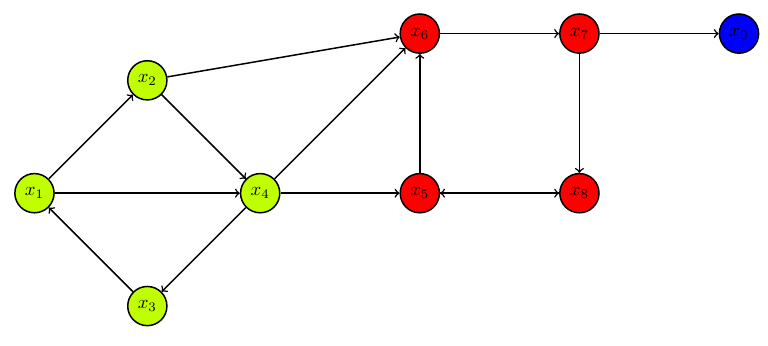
Obrázok, na ktorom je hodinky

Automaticky generovaný popis

Obrázok 3 Nesúvislý graf (naľavo) a súvislý graf (napravo)

**Silný komponent grafu**:

Silný komponent orientovaného grafu je maximálny silný podgraf grafu . Ak sú všetky silné komponenty orientovaného grafu tak potom . Orientovaný graf s jedným vrcholom je taktiež silný. Naviac musí platiť pre všetky , ináč by všetky vrcholy z množiny boli navzájom dosažiteľné a teda vrcholy by patrili do rovnakého silného komponentu. [Pócs, 2021]



Obrázok 4 Jednotlivé farby vyznačujú silné komponenty grafu.

# Najkratšia cesta

Hľadanie najkratšej cesty medzi dvoma (alebo viacerymi) vrcholmi v grafe je jedna z najzakladnejších algoritmických úloh v teori grafov, ktorá má svoje praktické využitie ako napr. hľadanie najkratšej cesty z mesta A do mesta B, doručenie paketov cez internetovú sieť ku koncovému prijimateľovi a pod. Medzi jedny z najznámejších algoritmov, ktoré vyhľadavaju najkratšiu cestu v ohodnotených orientovaných grafoch je Dijkstrov algoritmus, ktorý avšak pre svoje správne fungovanie potrebuje kladne ohodnotené hrany ďalej pozname algoritmus Bellmana a Forda, ktorý oproti Dijkstrovému je o nieco pomalši avšak dokáže pracovať s negatívne zapornými hranami.

**Definícia**:

V orientovanom grafe definujeme váhu cesty ako súčet váh hrán ktorými cesta prechádza.

Ďalej definujeme **váhu najkratšej cesty** z vrcholu do vrcholu ako:

Potom **najkratšia cesta** v ohodnotenom grafe je zadefinovaná ako akákoľvek cesta s váhou [Pocs,2021] [Cormen, Leiserson, Rivest, Stein, 2009]

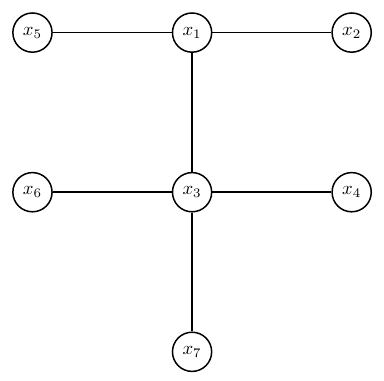
Obrázok 4 Červenou farbou je vyznačená najkratšia cesta z  do

# strom, les a kostra grafu

Strom v informatike je často používaná dátová štruktúra, ktorej rôzne vylepšené modifikácie nám dovoľujú rýchlo ukladať a vyhľadávať dáta. Medzi najznámejšie modifikácie stromov patrí napr. **červeno-čierny strom**, ktorý patri medzi binárne vyhľadávacie stromy, používaný napr. v kerneli operačného systému Linux.[Landley, 2007] Ďalšou známou modifikáciou je aj vyvážený n-árny koreňový **B-strom,** ktorý sa používa pri implementácií rôznych typov indexovania v databázach. [Drlík, Skalka, 2007].Pojem stromy neni neznámy ani v teórií grafov kde reprezentujú veľmi jednoduché grafy, ktoré aj napriek ich jednoduchosti tvoria zaujímavú oblasť štúdia a práve jeden typ týchto stromov sa nazýva aj **kostra grafu.** Pomocou tohto stromu dokážeme riešit problemy ako napr. lepšie navrhnutý vodovodný system alebo sledovanie histórie prenosu infekčných ochorení od ich prepuknutia. [Needham, Hodler, 2019]

**Strom**:

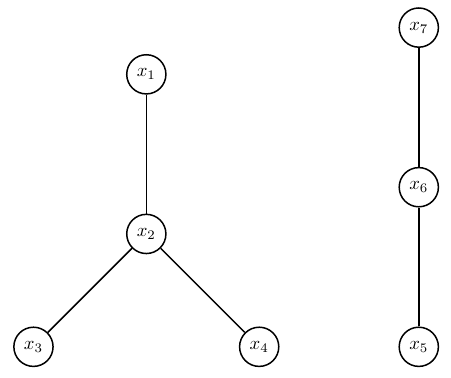
Súvislý neorientovaný graf bez kružnice. [Tomanová, ROK]



Obrázok 5 Strom

**Les**:

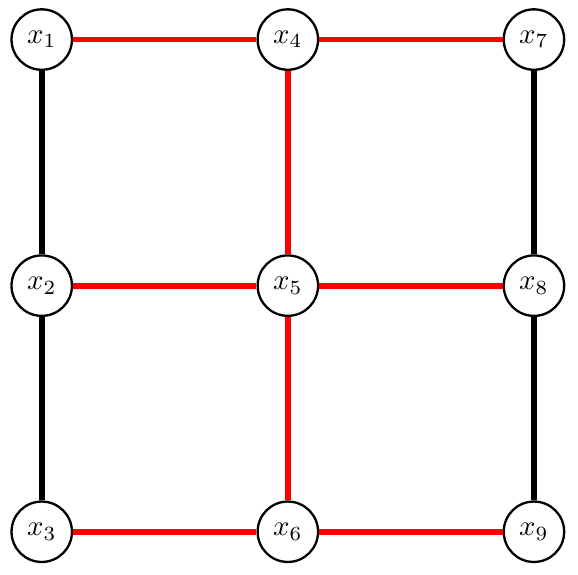
Graf bez kružníc, jednotlivé komponenty grafu sú stromy. [Hliněný, 2010]



Obrázok 6 Les

**Kostra grafu:**

Ak máme súvislý neorientovaný graf , je jeho faktor a ak je zároven aj stromom, tak hovoríme, že je **kostra grafu .** Každý neorientovány súvisly graf má aspoň 1 kostru.[Nečas, 1978]. V ohodnotených neorientovaných grafoch rozlišujeme maximálnu a minimálnu kostru. Palúch (2008) uvádza najdrahšiu kostru **(maximálna kostra)** v grafe s najväčšou cenou a najlacnejšiu kostru **(minimálna kostra)** v grafe zas s najmenšou cenou kde **cena kostry** je súčet ohodnotení jej hrán.



Obrázok 7 Červené vyznačený podgraf reprezentuje kostru

# Algoritmy

## Prehľadávanie do ŠÍRKY

Prehľadávanie do šírky (Breadth-first search) ďalej už iba BFS, patrí medzi jeden z najľahších prehľadavacích algoritmov, ktorý slúži ako základný kameň pre komplexnejšie algoritmy ako je napr. Primov algoritmus pre hľadanie minimalnej kostry v grafe [Cormen, Leiserson, Rivest, Stein, 2009].

**Postup:**

Tento algoritmus prehľadáva vrcholy tak, že najprv odhalí všetkých následovnikov počiatocneho vrcholu ale s tým, že si pamätá v akom poradí ich ohalil, toto zapamätanie môžme z programatorského hladiska zaistiti použitím dátovej štruktúry nazývanej fronta, do ktorej postupne vkladáme odhalené vrcholy a keď sú všetci nasledovníci odhalení tak zo začiatku fronty si vybereme ďaľší vrchol a algoritmus opakujeme. Toto odhaľovanie skončí ak sú všetky dosažitelné vrcholy odhalené, inač povedané ked fronta bude prázda. Algoritmus dokážeme upravit aj tak aby sa odhalil celý graf a nie len z počiatocného vrcholu [Cormen, Leiserson, Rivest, Stein, 2009].

**Atribúty vrcholov**:

Vrcholy obsahujú nasledujúce atribúty:

* *v.pi –* vrchol z ktorého bol vrchol *v* objavený (predchodca). Na začiatku majú všetky vrcholy nastavenú hodnotu na *None*
* *v.d –* vrstva*,* vrámci ktorej bol daný vrchol *v* odhalený. Pri inicializácií majú všetky vrcholy nastavené na *Nekonečno,* počiatočny vrchol ma 0.
* *v.color –* Šedá farba reprezentuje, že dany vrchol bol objavený, biela farba, že vrchol nebol ešte objavený a čiernu farbu má vrchol, ktorý už bol preskúmaný. [ZDROJ]

**Zložitosť:**

Asymptotická zložitosť BFS algoritmu je (kde  je množina vrcholov a

je množina hrán) lebo v najhoršom prípade bude muset algoritmus prejsť každou jednou hranou a každým vrcholom grafu [Cormen, Leiserson, Rivest, Stein, 2009].

**Použitie v realnom svete**:

BFS algoritmus väčšinou tvorí základ pre algoritmy zamerané na konkrétny cieľ ako napr. hľadanie najkratšej cesty. Určitá modifikácia algoritmu je použitá aj pre garbage collection metódu na spravovanie pamäti [Cheney, 1970].

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok 8 BFS pseudokód[[1]](#footnote-1)

## Prehľadávanie do hĺbky

Prehľadávanie do hĺbky (Depth-first search) ďalej už iba DFS, je ďaľším prehľádavacím algoritmom. DFS je všeobecne uznaný ako veľmi silný algoritmus pre riešenie rôznych grafových algritmov. Avšak algoritmus bol známy od 19. storočia ako technika na hľadanie cesty v grafe. [Even, 2012]

**Postup**:

DFS algoritmus objavuje hrany, ktoré vychádzajú z naposledy odhaleného vrcholu *v.* Keď algoritmus objaví koncový vrchol, ktorý ešte nebol odhalený, označí ho za objavený. Ak všetky hrany z naposledy odhaleného vrcholu *v* boli objavené tak „backtracking“ metódou sa vráti naspät do vrcholu, z ktorého bol vrchol *v* objavený a algoritmus sa opakuje pokiaľ sa neodhalia všetky dosažitelné vrcholy z počiatočného vrcholu [Cormen, Leiserson, Rivest, Stein, 2009].

**Zložitost**:

Asymptotická zložitosť DFS algoritmu je [Černá, 2019].

**Použitie v realnom svete**:

DFS algoritmus môžme použit napr. na hladanie cesty v bludisku alebo na nájdenie **topologického usporiadnia** orientovaných acyklických grafoch.

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok 9 DFS pseudokód[[2]](#footnote-2)

## BEllman-ford algortimus

BFS a DFS algoritmy fungovali na neohodnotených grafoch ale modelovať svet bez ohodnotených hrán by bolo priveľmi tažke ba až nemožné. Existuje niekoľko algoritmov, ktoré hľadajú najkratšiu cestu v ohodnotených orientovaných grafoch ale v tejto bakalárskej práci sme sa rozhodli popísat Bellman-Ford algoritmus, ktorý síce je oproti Dijkstrovému algoritmu pomalší avšak dokáže pracovať so zápornými hranami a detekovať negatívne cykly.

**Postup**:

Bellman-Ford algoritmus hľadá najkratšiu cestu z počiatočného vrcholu do všetkých ostatných vrcholov. Majme orientovaný ohodnotený graf  s pociataočným vrcholom *s*afunkciu *.* Algoritmus vracia booleanovskú hodnotu, ktorá nám hovorí či existuje negatívny cyklus, ktorý je dosiahnuteľný z počiatočného vrcholu. Ak taký cyklus neexistuje algoritmus vytvorí najkratšie cesty a vráti hodnotu **True**.

Algoritmus vyuziva metodu relaxaciu hran ktorou postupne znizuje hodnotu *v.d*  vsetkych vrcholov na hodnotu najkratsej cesty z pociatocneho vrcholu az pokial sa nedosiahne najkratsia cesta [Cormen, Leiserson, Rivest, Stein, 2009].

**Atributy vrcholov:**

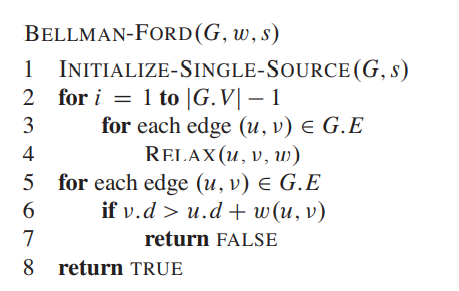
* *v.pi –* vrchol z ktorého bol vrchol *v* objavený (predchodca). Na zaciatku majú všetky vrcholy nastavené na *Nil*
* *v.d –*  vzdialenosť (distance)

**Zlozitost:**

Asymptotická zložitosť algoritmu je ale pri určitom type grafu dokážeme túto zložitosť zlepšit. Pri orientovaných acyklických grafoch si dokážeme zistit topologické usporiadanie a podľa tohoto usporiadania postupne relaxovať hrany v takom poradí akom sa vyskytujú na ceste. Takto dokážeme znížit asymptotickú zložitosť algoritmu na [Černý, 2010].

**Použitie v realnom svete**:

Rôzne modifikácie algoritmu sa využívajú napr. v smerovacom protokole **RIP (**routing information protocol**)** [Čipka, 2007].



Obrázok 10 Bellman-Ford pseudokód[[3]](#footnote-3)

## KRuskalov algoritmus

Vstupný grafmôže obsahovať veľa minimálnych kostier a nájsť tú najlepšiu z nich sa môže zdať ako obtiažna úloha avšak tento problém je algoritmicky veľmi dobre zvládnutelný pomocou Kruskalovho algoritmu, ktorý sa radi medzi „pažravé“ algoritmy. Tieto algoritmy sa vyznačujú tým, že vyberajú vrámci aktuálneho stavu vždy tu nejlepšiu možnosť [ Matoušek a Nešetril, 2007].

**Postup**:

Matoušek a Nešetril (2007) opisujú Kruskalov algoritmus nasledovne:

Majme súvislý ohodnotený graf s *n* vrcholmi, *m* hranami a s ohodnotením *w*.

Očíslujeme si hrany a zoradíme ich tak aby .

Tento krok si vyžaduje použit jeden z triediacich algoritmov idealne s najlepšou zložistosťou. Ak hrany budú zoradené budeme postupne konštruovať množiny hrán . Položíme . Algoritmus pokračuje ďaľej tak, že ak bola už nájdená množina tak množinu nasledujúco:

Algoritmus skončí ak obsahuje hrán alebo teda všetky hrany grafu sú spracované.Výsledkom bude podgraf , kde značí množinu, pre ktoru sa algoritmus zastavil.

**Zložistosť**:

Algoritmus začína zotriedením všetkych hrán vzostupne, čo mu trvá .

Ďalej predpokladáme, že používame dátovú štruktúru **disjoint-set**, v ktorej vykonáme-krát operáciu **FIND** a ()-krát operáciu **UNION**. To mu zabere čas . Jednotlivé operácie dokážeme upraviť tak aby čas trval iba kde je inverzná Ackermanová funkcia avšak celkovú zložitosť to nezmení [Černý, 2010].

**Použitie v reálnom svete**:

Výsledkom Kruskalovho algoritmu je minimálna kostra, ktora napr. môže byť použitá v ekonomickej sfére na vizualizáciu korelácií medzi menovými výnosmi [Needham, Hodler, 2019].

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok 11 Kruskal pseudokód[[4]](#footnote-4)

## Primov Algoritmus

Ďaľším algoritmom čo hľada minimalnú kostru podobne ako Kruskalov algoritmus je Primov algoritmus ktorý sa niekedy nazýva aj ako DJP algoritmus kvôli tomu že tento algoritmus bol trikrát nezávisle objavený prvykrát V. Jarníkom v roku 1937, R. C. Primom v roku 1957 a neskôr aj E.W.Dijsktrom v roku 1959. Tento algoritmus sa taktiež radi medzi „pažravé“ algoritmy.

**Atributy vrcholov:**

* *–* reprezentuje rodiča vrcholu v strome.
* *–*  najmenšia váha hrany, ktorá je spojená s vrcholom a vrcholom, ktorý sa nachádza v strome.

**Postup**:

Algoritmus vychádza z jediného podstromu , ktorý na začiatku obsahuje ľubovoľný počiatočný vrchol a počas behu algoritmu je podstrom postupne doplnený o takú hranu, pre ktorú platí, , a váha je minimálna. [Kolář, Štěpánková, Chytil, 1989].

**Žložistosť**:

Časová zložitosť závisí od toho ako implementujeme **prioritnú frontu** , pomocou ktorej dokážeme rýchlejšie vyberať vrcholy na základe atribútu. Ak použijeme binárnu haldu tak zložitosť je , při použití Fibbonaciho haldu zložitosť je .

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok 12 Prim pseudokód[[5]](#footnote-5)

**Použitie v reálnom svete:**

Primov algoritmus produkuje minimálnu kostru, ktorá môže byť použita pri minimalizácií cestovných nakladov při spoznavaní krajiny [Needham, Hodler, 2019].

## Tarjanov algoritmus

Robert Tarjan v roku 1972 vytvoril algoritmus, ktorý dokáže nájsť silne súvislé komponenty, vďaka upravenému DFS algoritmu.

**Postup**:

Algoritmus vybere ľubovolný vrchol a priradime mu poradové čislo , následne mu priradíme najmenšie poradove číslo a nakoniec vložíme do zásobníku. Pokračujeme v DFS preskúmavaní a každému novému nepreskúmanému vrcholu nastavíme poradové číslo a najmenšie poradové číslo. Ak je vrchol už preskúmaný ale nie je zaradený do silného komponentu tak skontrolujeme, či . Ak áno tak , vložíme do zásobníka a prehľadávame ďalej. Môže sa stať, že pri prehľadávaní narazíme na vrchol ktorý už patri do silného komponentu, tak potom vyberáme nasledujúci vrchol v poradí a pokračujeme v prehľadávaní. Pri “backtracking” metóde môžme narazit na vrchol , pri ktorom , potom vieme že sme našli silnu komponentu, ktorá bude obsahovať doteraz nezaradené vrcholy , pre ktoré plati . Vrcholy komponentu vymažeme zo zásobníku. Ak je zásobník prázdny vyberáme ďalší nepreskúmaný vrchol a algoritmus opakujem. Pri neprázdnom zásobníku, označíme vrchol zásobníku a ak nebola uzavretá komponenta a položíme , a pokračujeme v prehľadávaní. Algritmus skonci ak všektky vrcholy budu odhalené [Demlová, 2018].

**Zložitosť:**

Asymptotická zložitosť Tarjanovho algoritmu je lineárna, čiže [Demlová, 2018].

**Použitie v reálnom svete:**

Silne súvislé komponenty môžeme nájsť v rôzných sociálnych sieťach, kde jednotlivé komponenty reprezentujú určitú množinu ľudí na základe ich společných preferencií a vďaka algoritmom, ktoré hľadaju komponenty môžme týmto ľudom cielene odporúčit stranku, ktoru by mali “likúť” alebo produkt, ktory by si mali zakúpit [Needham, Hodler, 2019].

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok 12 Tarjan pseudokód[[6]](#footnote-6)

## TopologickÉ Radenie

Existuje veľa situácií pri ktorých záleži v akom poradí vykonáme jednotlivé úlohy. Zoberme si príklad študenta vysokej školy, ktorý músi vziať do úvahy, ktoré predmety sú prerekvizitou alebo nutnou podmienkou pre ostatné predmety pri zostavovaní rozvrhu na semester. Takýto problém závislosti môžeme reprezentovať iba na acyklickom orientovanom grafe a vďaka topologickému radeniu dokážeme nájsť správne poradie ako si predmety zapísať.

Černý (2010) definuje topologické usporiadanie ako usporiadanie vrcholov do rady tak, že každá hrana vedie zľava doprava. Inak povedané je to také očislovanie vrcholov 1 až , kde každá hrana vedie z vrcholu s menším číslom do vrcholu s väčším číslom.

**Atributy vrcholov:**

* *–* udáva čas kedy bol prieskum vrcholu ukončený.

**Postup**:

Hľadanie topologického usporiadania a rozpoznávať či je graf acyklický vieme spojit do jedného DFS priechodu. Spustíme DFS prehľadávanie a ak nájdeme spätnú hranu tak algoritmus môžeme ukončiť lebo žiadne usporiadanie neexistuje, inak vypíšeme vrcholy podľa atribútu vzostupne [Černý, 2010].

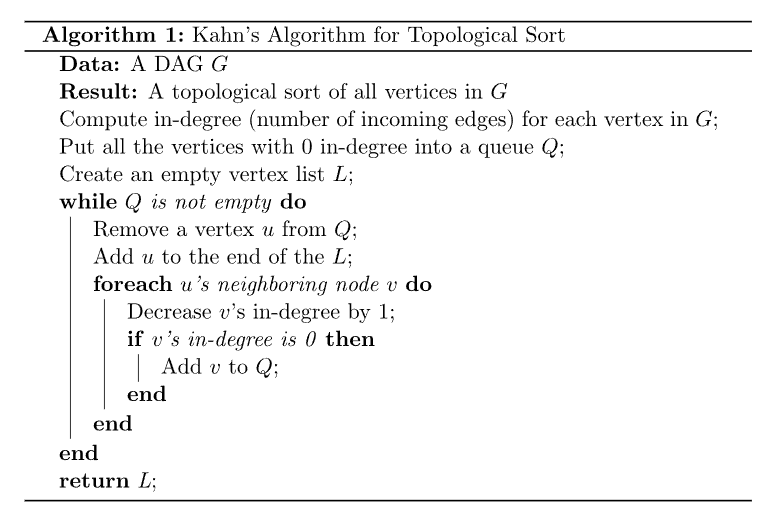
Ďalšou alternatívou je postupne odstraňovanie a vypisovanie vrcholov do ktorých nevedie žiadna hrana. Pokiaľ odstraníme vrchol musíme odstránit všetky jeho hrany a algoritmus opakujeme, tento algoritmus sa nazýva **Kahnov algoritmus** [Kahn, 1962].

**Zložitosť**:

Zložitosť prvého spomenutého postupu za pomoci DFS priechodu je [Černý, 2010]. Kahnov algoritmus ma zložitosť taktiež [Kahn, 1962].

**Použitie v reálnom svete:**

Niekedy program pre svoje správne fungovanie potrebuje spustit niekoľko zdrojových suborov v správnom poradí. Takto približne funguje **Makefile** čo je konfiguračný súbor so závislostami pre unixový program **make**. Jednotlivé závislosti napíšeme do súboru Makefile a podľa tohoto súboru **make** spusti použite programy v správnom poradi. Takéto usporiadanie nazýváme aj ako **topologické usporiadanie** [Černý, 2010].



Obrázok 13 Kahn pseudokód[[7]](#footnote-7)

# postup vytvárania webovej aplikácie

Vrámci praktickej časti tejto bakalárskej prace sme sa rozhodli naprogramovať webovú aplikáciu, pre vizualiazáciu grafov a jednotliavých algoritmov z teórie grafov. Webová aplikácia, ďalej už len aplikácia, je naprogramovaná pomocou technológií zameraných na tvorbu webových stránok a to **HTML**, **CSS** a **Javascriptu**. Avšak pre lepší a rýchlejší vývoj aplikácie sme použili a knižnicu **Jquery** a framework **Bootstrap**. Aplikácia sa samozrejme nezaobisla bez **Cystoscape** knižnice, ktorá tvorí jadro samotnej vizualizácie. Obe knižnice sme nainštalovali pomocou package manageru **npm**, ktorý je súčasťou prostredia **Node.js**. Framework Bootstrap sme implementovali do aplikácie pomocou **CDN** (Content delivery network). Samotná aplikácia bola následne nahraná na webovú doménu <https://grapher.demi.sk/> a zdrojový kód je dostupný na webovej službe GitHub <https://github.com/LukasHajda/Bachelor_project> .

**Cytoscape knižnica**:

Cytoscape je nezávislá Javascriptovská knižnica, ktorá na svoje správne fungovanie nepotrebuje žiadne dodatočne rozširenia, pluginy do samotného browseru alebo iné knižnice. Architektúra Cytoscape.js dovoluje jeho spustenie bez ďalšieho grafickeho pouzivatelského rozhrania (GUI) kde základ imeplemntacie stoji na využiti plátna (canvas) v HTML. Knižnica zdieľa niekoľko konceptov z HTML, CSS a samotného JS. Štýlovanie v Cytoscape.js pomocou špecifickej syntaxie podobnej tej z CSS [Franz, Lopes, Huck, Dong, Sumer ,Bader, 2015].

**Vlastnosti:**

Cytoscape.js poskytuje programátorovi niekolko funkcionalít, ktore môže využit a to:

* **Typy grafov**: Cytoscape.js podporuje vizualizovať niekoľko typov grafov ako napr. orientované grafy, neorientované grafy, ohodnotené, neohodnotené.
* **Import a export**: Vytvorený graf je možné si vyexportovať v obrázkovom formáte PNG alebo JPG vo vysokej kvalitne. Taktiež podporuje import / export grafu vo forme JSON formátu.
* **Rozloženie**: Cytoscape.js obsahuje rôzne typy rozloženia grafu, ktoré každé z nich dovoľuje iné iné rozloženie vrcholov na plátne.
* **Rozšíritľnosť**: Poskytuje mechanizmus pre programátora na rozšírenie funckionalit knižnice [Franz, Lopes, Huck, Dong, Sumer ,Bader, 2015].

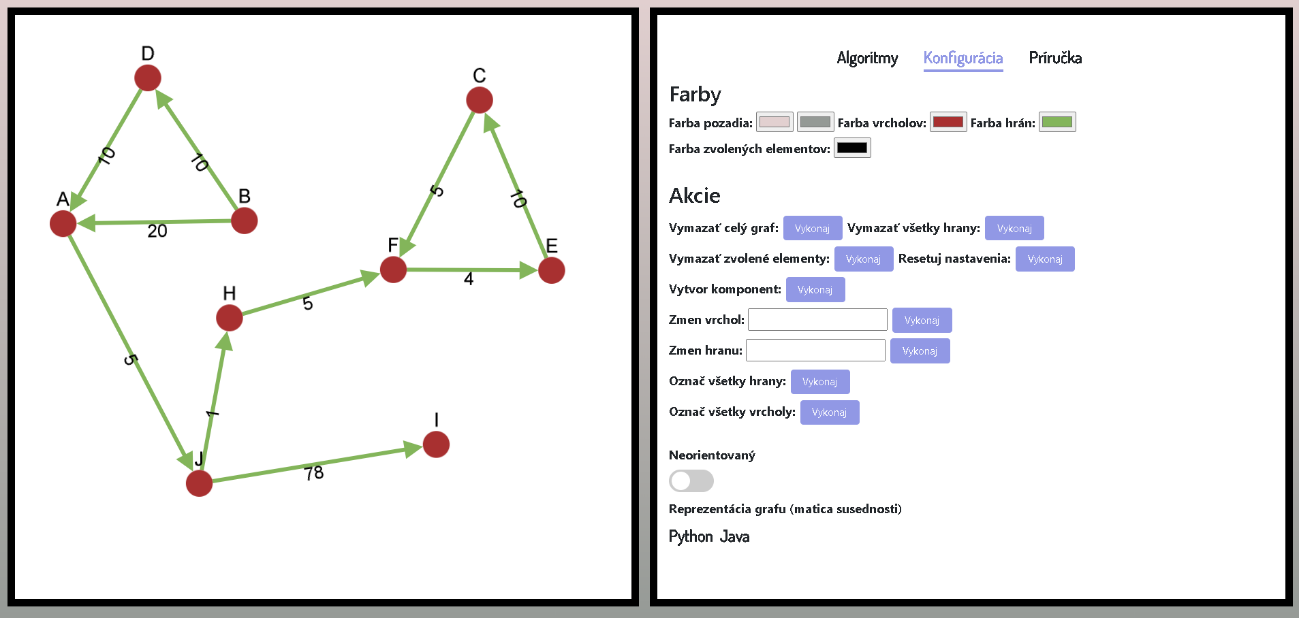
**Výkonnosť**:

Cytoscape.js zvládne bez problémov vykresliť tisíc prvkov aj pri slabšom hardveri. Samozrejme výkon vykreslovania značne ovplyvňuje množstvo použitých vizuálných štýlov a aj samotný internetový prehliadač. Cytoscape.js je šetrný pri vykreslovaní a preto nový snimok sa vykreslí iba ak je to potrebné [Franz, Lopes, Huck, Dong, Sumer ,Bader, 2015].

## Štruktúra Aplikácie

Aplikácia je z používateľského pohľadu rozdelená na 2 časti, kde prvá časť (Obrázok 14 ľavá časť) je samotné plátno kde si používateľ vytvára graf a kde pribieha samotná vizualizácia zvoleného algoritmu.

Drúha časť (Obrázok 14 pravá časť) predstavuje menu, ktoré obsahuje 3

****

Obrázok 4

## 1.1 Okraje dokumentu

Odsadenie textu od **ľavého okraja** strany dokumentu je v celej ZP nastavené na **3,5** cm, kvôli viazaniu ZP.  **Ostatné** okraje sú nastavené štandardne na **2,5** cm.

## 1.2 Odsadenie odsekov

Vzájomné odsadenie odsekov v texte ZP je nastavené už v šablóne ZP, pomocou **štýlov nadpisov**, kde sa nachádzajú už prednastavené štýly i medzery nad a pod nadpismi. Žiadne iné odsadenie medzi odsekmi sa nepoužívajú. Žiadne zbytočné medzery, tvorené ENTEROM, nie sú žiadúce.

## 1.3 číslovanie a ODRÁŽKY

Na vkladanie číslovaných zoznamov sa používa číslovanie a pre nečíslované zoznamy odrážky. Najrýchlejšie sa vkladajú a odstraňujú pomocou ikon na karte *Odsek.* Týmito tlačidlami pridáme alebo odoberieme nastavený typ číslovania alebo odrážok.

1. Znak odrážky je zarovnaný od ľavého okraja bez ďalšieho odsadenia.
2. Text za odrážkou je odsadený o 1,2 cm (ako vidíte v tomto texte).
3. Vy budete používať v celej ZP znak odrážky ◾ - **čierny štvorček.**

## 1.4 typografické pravidlá - Dvojbodka, čiarky, bodka v texte za odrážkou

Ak v texte za **dvojbodkou,** v novom riadku**,** vkladáte odrážky alebo číslovanie, musí byť p**rvé písmeno** za odrážkou **malé. Za textom** odrážky musí byť **čiarka**. Za **posledným** textom odrážky musí byť vložená **bodka.** Takto vyzerá ukážka:

* toto je prvý text za odrážkou s čiarkou,
* toto je druhá odrážka s čiarkou,
* toto je posledná odrážka s bodkou.

## 1.5 uvádzanie citácií v texte

Uvádzanie citácie v texte je veľmi dôležité a musí presne zodpovedať všetkým citáciám uvedeným v bibliografických odkazoch na konci ZP. V texte sa môže uvádzať citácia rovnako v úvode vety alebo po ukončení myšlienky vo vete. Citácia v texte sa uvádza nasledovne – okrúhla zátvorka + priezvisko autora + rok vydania publikácie + okrúhla zátvorka a až potom nasleduje bodka – takto (Rozmarín, 2008).

Ak uvádzate **viacerých citovaných autorov** v odseku, tak to budete písať nasledovne (Rozmarín, 2008; Rozmarínová, 2016; Kovová, 2017; Hrdzavý a kol., 2016). Ako vidíte, jednotliví autori sa oddeľujú medzi sebou **bodkočiarkou**.

## 1.6 Ukážka zápisu citácie v texte

Nemusíte písať vždy autora citácie za textom v odseku. Môžete logickú nadväznosť textu uviesť (parafrázovať) aj takto:

* Rozmarín (2008) uvádza, že prírodné vedy ......text text text text text text text.
* Autorka (Kvetinková, 2016) predkladá názor, že prírodné vedy .......text text text.
* Prírodné vedy sú ...... text text text text, ako uvádzajú autori príspevku   
  (Hrdzavý a kol., 2016 – zápis do textu). Do Zoznamu bibliografie sa musí zapísať každý člen kolektívu).

## 1.7 Ukážka zápisu autorov v bibliografickom zozname

Pri citovaní používame jednotnú normu ISO-690 (<https://www.uk.ukf.sk/data/epca/harvard-iso690.pdf>), kde v texte sa používa zápis (Autor, rok). V záverečnej práci by ste mali vychádzať prevažne z článkov v elektronických časopisoch, ktoré sú evidované v citačných databázach Web of Science alebo Scopus. Prístup k elektronickým databázam máte zadarmo cez CVTI: <https://www.cvtisr.sk/sk/specializovana-vedecka-kniznica/kniznicno-informacne-sluzby/registracia-pouzivatelov-a-vydavanie-kniznicnych-pasov.html?page_id=551&preview_type=from_menu>.

Pozor pri zápise elektronických časopisov, nejedná sa o zdroj z internetu!

HRDZAVÝ, J. KVETINKOVÁ, V., ROZMARÍN, T. 2018. Modul-1. In : *Zborník príspevkov z informatického seminára KI. ISKI 2018*. UKF Nitra, 2018. s. 160-168. ISBN 978-80-8094-351-6. (Zápis z tlačeného zborníka, In: znamená, kde sa článok nachádza, v tomto prípade v Zborníku príspevkov....).

KVETINKOVÁ, V. 2016. *Teória vyučovania informatiky***.** [online] Tlačová správa, 2016. [cit. 2017-01-09]. Dostupné na internete <http://www2.statpedu.sk/buxus/ generatepage.php\_page\_id=649.htm l#pedagogi ckyproces\_kvetinkova>. (Zápis z internetovej stránky)

ROZMARÍN, T. 2008. *Nové trendy v prírodných vedách*. UKF Nitra, 2008. 265 s. ISBN 80-8069-677-2. (zápis z tlačenej knihy)

VAŘEKA, Pavel, 2010. Příspěvek ke studiu žijících vsí středověkého původu. Pozůstatky zástavby z pozdního středověku na parcele č.p. 121 v Mikulčicích. *Přehled výzkumů*. **51**(1-2), 249-265. ISSN 1211-7250. (zápis článku z časopisu v licencovaných databázach, ak má článok doi číslo, tak uvádzať radšej doi, než ISSN)

# 2 CIELE záverečnej práce

Druhá kapitola sa venuje cieľom/cieľu ZP. Cieľom je predovšetkým preukázať schopnosť preštudovať nie veľmi širokú a nie veľmi úzku problematiku. Spracovať ju do kompaktného, vysvetľujúceho textu, ktorý je jednoznačnou reakciou na vopred stanovený cieľ. Ten je zvyčajne charakterizovaný problémom. Cieľ stanovuje pedagóg presne špecifikovanou anotáciou k téme ZP. Vy musíte jednoznačne a odborne naformulovať hlavný cieľ ZP.

## Podciele alebo čiastkové ciele:

* predstavujú jednotlivé kroky, ktoré si musíte stanoviť na dosiahnutie hlavného cieľa ZP práce (používajú sa činnostné slovesá napr. analyzovať, porovnať, navrhnúť, namodelovať, spracovať, otestovať, vyhodnotiť, odporučiť pre prax),
* byť schopný nájsť relevantné informačné zdroje v optimálnom rozsahu,
* byť schopný vybrať, vytriediť relevantné informačné zdroje z dostupných databáz odborného charakteru,
* interpretovať odborným štýlom (nie novinárskym alebo silne neodborným „kuchynským“ štýlom),
* vedieť primerane odborne, jednoznačne a vedecky stroho formulovať svoje myšlienky,
* dokázať ozrejmiť si význam parafrázy a citácie a naučiť sa ich adekvátne používať,
* vedieť štylizovať tak, aby bol text pre čitateľa ľahko prístupný a zároveň zaujímavý a pochopiteľný.

# 3 Metódy / postupy riešenia ZPkeho projektu

**Pracovný názov kapitoly,** slúži vám ako návod. Vy si zadajte vlastný názov, po konzultácii so školiteľom ZP. Úvodné slovo k metódam riešenia problému. Metodika práce musí obsahovať: charakteristiku objektu skúmania, pracovné postupy, spôsob získavania údajov a ich zdroje, použité metódy vyhodnotenia a interpretácie výsledkov.

## 3.1 Táto časť ZP má obsahovať:

* opis metódy riešenia problému,
* zdôvodnenie voľby spôsobu riešenia v súlade s analýzou problému - môžete sa sústrediť napr. na prípadné modifikácie použitých štandardných metodík a ich zdôvodnenie z hľadiska splnenia cieľov práce,
* opis všetkých etáp riešenia projektu - v prípade, že projekt nerieši všetky etapy, **musí** byť v príslušnej časti uvedené kto, resp. **kde** **sa príslušná etapa rieši/riešila/bude riešiť.**

### Pri opise postupu riešenia treba zachytiť:

* všetky hlavné body riešenia a potom popísať postupné kroky riešenia,
* najmä zvláštnosti,
* úspechy a problémy pri riešení,
* neúspešné pokusy také, ktorých poznanie môže skvalitniť a alebo zefektívniť riešenie podobných projektov, ale nie také, ktoré vyplývajú z hrubej neznalosti problematiky.

### 3.1.2 Obrázky, tabuľky a grafy

**Obrázky** a **grafy** musia byť v texte centrované (Obrázok 1, Graf 1). Titulok musí byť umiestnený **POD** obrázkom alebo **POD**  grafom a centrovaný.

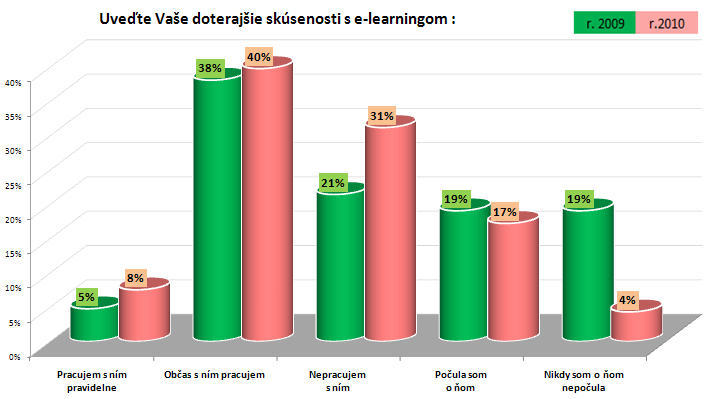
**Tabuľky** musia byť centrované a musia mať titulok **NAD** tabuľkou zarovnaný na ľavej strane s pozíciou tabuľky (Tabuľka 1 a 2).

Každý objekt v práci musí mať uvádzaný zdroj, pokiaľ nie je výtvorom autora záverečnej práce a **na každý obrázok, tabuľka a graf sa musíte v texte odkazovať**. Pozrite si nasledujúce ukážky.



Obrázok 1 Kriváň - Vysoké Tatry[[8]](#footnote-8)

Na vkladanie popisov a ich zoznamu, používajte automatické vkladanie cez príkaz Referencie/Vložiť popis.

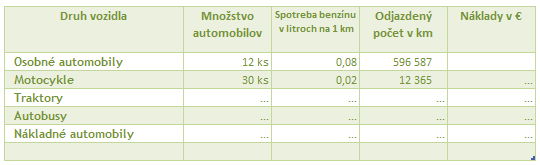


Graf 1 Skúsenosti učiteľov s e-learningom[[9]](#footnote-9)

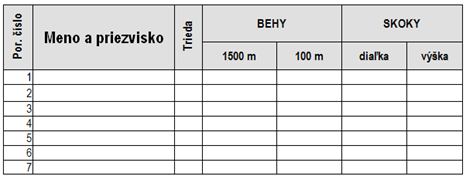
## 3.2 Zápis zdrojov tabuliek, grafov, obrázkov v texte

Zdroje tabuliek, grafov, obrázkov môžete vkladať ako *Poznámku pod čiarou*[[10]](#footnote-10) (Referencie/Poznámka pod čiarou).

*Tabuľka 1 Výpočet spotreby benzínu[[11]](#footnote-11)*



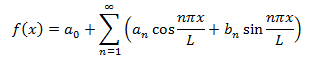
*Tabuľka 2 Tabuľka behov a skokov[[12]](#footnote-12)*



Na vkladanie popisov a vygenerovanie ich zoznamu používajte automatické vkladanie cez príkaz *Referencie/Vložiť popis.*

## 3.2.1 Rovnice v ZP

Rovnice v ZP sa uvádzajú rovnako ako tabuľky a ilustrácie. Musia byť centrované. Na vkladanie a editáciu rovníc je odporúčané použiť preddefinované matematické symboly, ktoré nájdete *Vložiť/Symboly/Rovnica.*



# 4 Výsledky riešenia a ich zhodnotenie

Vlastná časť ZP patrí medzi najvýznamnejšiu časť práce. Výsledky - vlastné postoje alebo vlastné riešenie vecných problémov, ku ktorým autor ZP dospel, musí **logicky** **usporiadať** a pri popisovaní fakty a poznatky dostatočne zhodnotiť a  komentovať. Vlastná práca je najrozsiahlejšou časťou práce a má predstavovať **30 %** ZP..

## 4.1 Časť Výsledky riešenia

V tejto časti je potrebné:

* vyhodnotiť potvrdené, vyvrátené alebo nepotvrdené hypotézy,
* keďže výsledky zahrnujú nové poznatky, ktorými autor prispieva k svetovému poznaniu, je dôležité, aby boli jasne a jednoducho definované, pritom však nie príliš stručné,
* dôležité je uviesť i prípadnú štatistickú významnosť nálezov a údajov; číselné údaje výsledkov sa uvádzajú, ak to netreba uviesť inak, s presnosťou na jedno, maximálne dve desatinné miesta,
* pri publikovaní výsledkov je vhodné, ak sa kombinuje text s tabuľkami, grafmi a obrázkami so zvýraznením údajov a trendov.

### 4.1.1 Časť Výsledky riešenia sa píše v minulom čase

Časť *Výsledky riešenia* sa píše v **minulom čase.** *Časť Výsledky riešenia sa píšu v minulom čase. Časť Výsledky riešenia sa píšu v minulom čase. Časť Výsledky riešenia sa píšu v minulom čase. Časť Výsledky riešenia sa píšu v minulom čase. Časť Výsledky riešenia sa píšu v minulom čase. Časť Výsledky riešenia sa píšu v minulom čase. Časť Výsledky riešenia sa píšu v minulom čase.*

# Záver

Záver ZP sumarizuje vlastný prínos, alebo pohľad autora na riešenú problematiku. Mal by obsahovať:

**stručné výsledky ZP, v súlade so stanoveným cieľom.**

V závere ZP sa odporúča:

* poukázať na vyriešené a nevyriešené problémy,
* naznačiť ich význam a možné riešenie,
* predstaviť návrh na využitie výsledkov.

Rozsah kapitoly ZÁVER je odporúčaný cca na **1** stranu.

# Zoznam bibliografických odkazov (ukážka, ale všetky uvedené zdroje musia mať odkaz na autora v texte).

HRDZAVÝ, J., HRDZAVÁ, K., NOVÝ, Z., STARÝ, L. 2016. *Nové trendy*. [online] 2016. [cit. 2019-02-14]. Dostupné na internete: <http://mcmb.mcmb.sk/ESF/ tvorba\_ekurzov.htm>.

KHAN, B.H. 2006. *E-learning - Osem dimenzií otvoreného, flexibilného a distribuovaného e-learningového prostredia*. SPU Nitra, 2006. 265 s. ISBN 80-8069-677-2.

PALMÁROVÁ, V. 2008. Teórie učenia a ich aplikácia v e-podpore vyučovania**.**   
In : *Zborník príspevkov z informatického seminára KI. ISKI 2008.* UKF Nitra, 2008. 160 s. ISBN 978-80-8094-351-6.

ŠPÚ. ŠTÁTNY PEDAGOGICKÝ ÚSTAV. 2011. Modul-1. [online] Tlačová správa, 2011. [cit. 2011-01-09]. Dostupné na internete: <http://www2.statpedu.sk/buxus/ generate page.php\_page\_id=649.html#pedagogickyproces>.

TURČÁNI, M. 2005. Learning - nová forma práce metodikov. Spracované z prednášky pre projekt *Tvorba e-kurzov.* [online]. 2011. [cit. 2011-02-14]. Dostupné na internete: [<http://mcmb.mcmb.sk/ESF/tvorba](http://mcmb.mcmb.sk/ESF/tvorba)\_ekurzov.htm>.

ZLÁMALOVÁ, H. 2008. Dištanční vzdelávaní a eLearning. UK Praha, 2008. 184 s. ISBN 978-80-86723-56-3.

VAŘEKA, Pavel, 2010. Příspěvek ke studiu žijících vsí středověkého původu. Pozůstatky zástavby z pozdního středověku na parcele č.p. 121 v Mikulčicích. *Přehled výzkumů*. **51**(1-2), 249-265. ISSN 1211-7250.

(ZORADIŤ PODĽA ABECEDY a NEČÍSLOVAŤ!)

# Zoznam príloh (ukážka - strany sa nepočítajú sa do ZP)

Príloha A – Projektová dokumentácia k ....(nová strana)

Príloha B – Zdrojový kód ....

Príloha C – Fotodokumentácia k prípravnej fáze projektu ........

PRÍLOHA A

1. Zdroj: Obrázok 8 : ZDROJ [↑](#footnote-ref-1)
2. Zdroj : Obrázok 9 : ZDROJ [↑](#footnote-ref-2)
3. Zdroj : Obrázok 10 : ZDROJ [↑](#footnote-ref-3)
4. Zdroj : Obrázok 11 : ZDROJ [↑](#footnote-ref-4)
5. Zdroj : Obrázok 11 : ZDROJ [↑](#footnote-ref-5)
6. Zdroj : Obrázok 11 : <https://www.programming-algorithms.net/article/44220/Tarjan's-algorithm> [↑](#footnote-ref-6)
7. Zdroj : Obrázok 11 : <https://www.baeldung.com/cs/dag-topological-sort> [↑](#footnote-ref-7)
8. Zdroj Obrázok 1: http://support.google.com/earth/bin/answer.py?hl=sk&answer=148146 [↑](#footnote-ref-8)
9. Zdroj Graf 1: https://www.digipay.sk/grafy-online/ [↑](#footnote-ref-9)
10. [↑](#footnote-ref-10)
11. Zdroj Tabuľky 1: http://edu.ukf.sk/course/view.php?id=652 [↑](#footnote-ref-11)
12. Zdroj Tabuľky 2: http://www.ki.fpv.ukf.sk/ [↑](#footnote-ref-12)