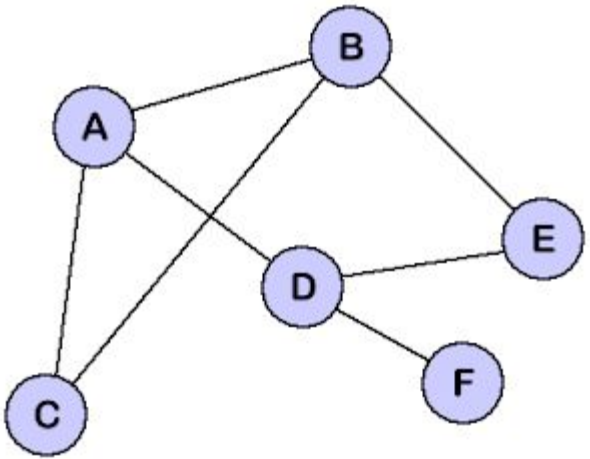
Grafy a Prehľadávacie algoritmy

Dátové štruktúry a algoritmy 24/25 LS
Prednášky, garant: prof. Gabriel Juhás
Cvičenia: Milan Mladoniczky - milan.mladoniczky@paneurouni.com

Graf

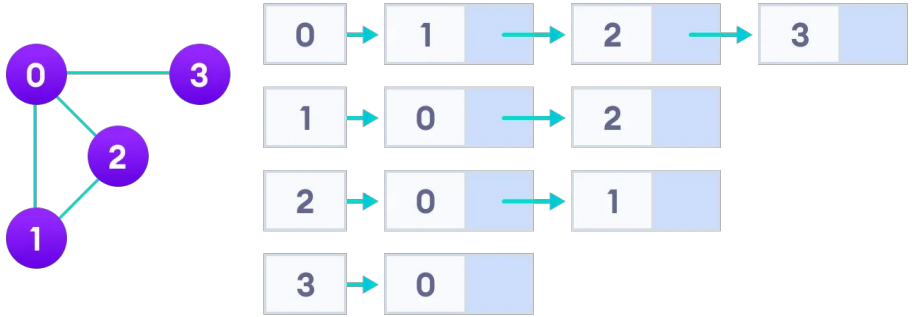
- Nelineárna dátová štruktúra pozostávajúca z hrán a vrcholov (uzlov).
- Kde vrchol môže byť spojený v 0-N inými uzlami hranami.
- Grafy môžu byť orientované, kedy hrana má definovaný smer (napr. pri analýze toku)
- Hrany môžu mať priradenú hodnotu, vtedy hovoríme o ováhovanom grafe.
- Modelovanie komplexných vzťah a súvislostí.



Funkcia	Graf	Strom
Definícia	Kolekcia uzlov (vrcholov) a hrán, kde hrany spájajú uzly.	Hierarchická dátová štruktúra pozostávajúca z uzlov spojených hranami s jediným koreňovým uzlom.
Štruktúra	Môže obsahovať cykly a nesúvisiace komponenty.	Nemá cykly; súvislá štruktúra s presne jednou cestou medzi akýmikoľvek dvoma uzlami.
Koreňový uzol	Nemá koreňový uzol; uzly môžu mať viacerých rodičov alebo žiadneho.	Má určený koreňový uzol, ktorý nemá rodiča.
Vzťah medzi uzlami	Vzťahy medzi uzlami sú ľubovoľné.	Vzťah rodič-dieťa; každý uzol (okrem koreňa) má presne jedného rodiča.
Hrany	Každý uzol môže mať ľubovoľný počet hrán.	Ak je n uzlov, potom je počet hrán n-1.
Zložitosť prehľadávania	Prehľadávanie môže byť zložité kvôli cyklom a nesúvislým komponentom.	Prehľadávanie je jednoduché a môže sa vykonať v lineárnom čase.
Použitie	Používa sa v rôznych scenároch, ako sú sociálne siete, mapy, optimalizácia sietí atď.	Bežne sa používa na reprezentáciu hierarchických dát, ako sú súborové systémy, organizačné diagramy, HTML DOM, XML dokumenty atď.
Príklady	Sociálne siete, cestné siete, počítačové siete.	Súborové systémy, rodokmene, HTML DOM štruktúra.

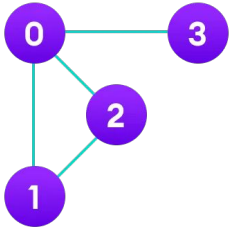
Graf ako zreťazený zoznam

- Jednoduchá implementácia na pochopenie.
- Náročnejšia na veľkosť pamäte.
- Použité pre vyhľadávacie algoritmy a grafové databázy.



Graf ako matica

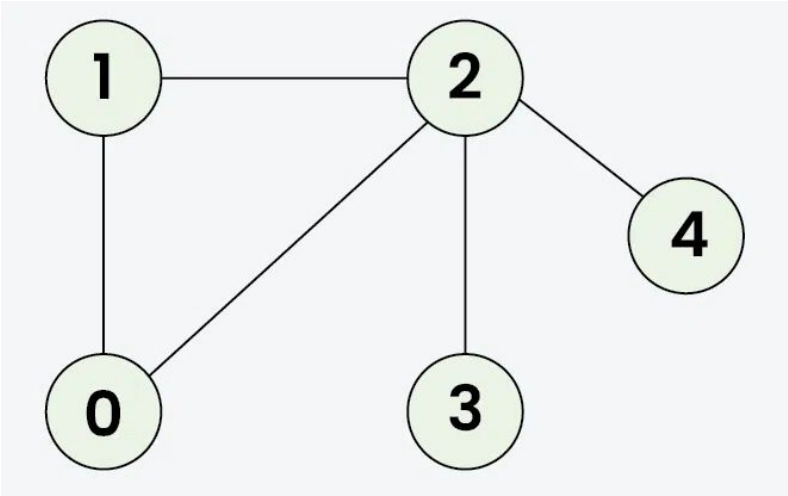
- Graf je možné implementovať ako maticu.
- Kompaktné implementácie aj väčších grafov.
- Použité hlavne pri algoritmických spracovaniach, data science, či AI.



	0	1	2	3
0	0	1	1	1
1	1	0	1	0
2	1	1	0	0
3	1	0	0	0

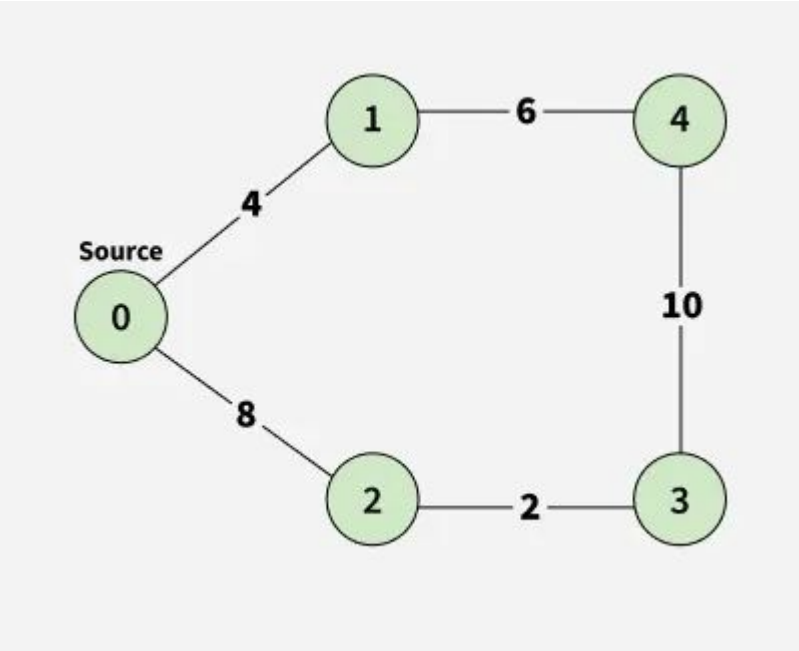
Prehľadávanie do hĺbky

- Prejdenie grafu po všetkých vrcholoch.
- Podobne ako pri strome avšak keďže graf môže obsahovať cykly musí byť držané dočasná kolekcia navštívených vrcholov.
- Keďže v grafe nie je vrchol ako root je potrebné ho určiť ako vstupný parameter.



Dijkstrov algoritmus najkratšej cesty

- Často používaný algoritmus na vyhľadanie najkratšej cesty medzi dvomi vrcholmi grafu.
- Pri neováhovaných grafom berieme, že každá hrana má váhu 1.
- Možné použiť aj pri orientovaných grafoch.
- Možné použiť iba pri nezáporných váhach hrán.



Dijkstra

- Keďže zdrojový vrchol je východiskovým bodom, jeho vzdialenosť je inicializovaná na nulu.
- Odtiaľto iteratívne vyberáme nespracovaný vrchol s minimálnou vzdialenosťou od zdroja, tu sa kvôli efektivite zvyčajne používa mini-heap (priority queue) alebo set.
- Pre každý vybraný vrchol *u* aktualizujeme vzdialenosť k jeho susedom v podľa vzorca:
  dist[v] = dist[u] + weight[u][v]
  ale len vtedy, ak táto nová cesta ponúka kratšiu vzdialenosť ako aktuálne známa.
- Tento proces pokračuje, kým sa nespracujú všetky vrcholy.

1. Nastavte *dist=0* a všetky ostatné vzdialenosti ako nekonečno.
2. Vložte zdrojový vrchol do heap ako dvojicu *<vzdialenosť, vrchol>* → <0, zdroj>.
3. Vyberte vrchný prvok z heap (vrchol s najmenšou vzdialenosťou).
   3.1. Pre každého suseda aktuálneho vrchola:
   3.2. Vypočítajte vzdialenosť pomocou vzorca:
   dist[v] = dist[u] + weight[u][v]
   Ak je táto nová vzdialenosť kratšia ako aktuálna *dist[v]*, aktualizujte ju.
   Aktualizovanú dvojicu *<dist[v], v>* vložte do heap.
4. Opakujte krok 3, kým nie je heap prázdna.
5. Vráťte pole vzdialeností, ktoré obsahuje najkratšiu vzdialenosť od zdroja ku všetkým uzlom.

Poďme sa pozrieť na implementáciu ->

https://dsa.interes.group/exercises/exercise-6