Algoritmizace - Grafy, Prohledávání stavového prostoru, Řazení

# Teorie grafů

Grafem rozumíme datovou strukturu, která popisuje vztahy mezi objekty.

Grafy z oblasti teorie grafů mají široké využití v mnoha oblastech, např. úlohy o dopravním spojení, logistické problémy, optimální spojení, propustnost sítě, přenos energie, komprese dat

Každý graf obsahuje:

* Body = vrcholy grafu (uzly)
* Linie = hrany graf

**Izolovaným vrcholem** je nazýván vrchol, který neinciduje žádná hrana grafu.

**Stupeň vrcholu** (též valence vrcholu) označuje počet hran, které do daného vrcholu zasahují. Stupeň vrcholu 𝑉 se značí deg(𝑉).

* orientovaná hrana – uspořádaná dvojice vrcholů; má vyznačen směr průchodu, hranou lze procházet pouze ve vyznačeném směru
  + Orientovaný graf je takový graf, jehož všechny hrany jsou orientované.
* neorientovaná hrana – neuspořádaná dvojice; bez vyznačení směru průchodu, hranou lze procházet oběma směry
  + Neorientovaný graf je takový graf, jehož všechny hrany jsou neorientované.
* násobné hrany – více hran spojujících stejné vrcholy
* most – hrana, jejímž odebráním se zvýší počet komponent grafu
* smyčka – hrana vedoucí z vrcholu do něj samotného
* jednoduchý (obyčejný) graf
* neobsahuje smyčky ani násobné hrany
* multigraf – obsahuje násobné hrany
* prostý graf – neobsahuje násobné hrany
* pseudograf – obsahuje smyčky

ohodnocený (každá hrana mezi dvěma uzli má nějakou hodnotu př: čas z jednoho uzlu do druhého) či neohodnocený graf

Obsah obrázku diagram

Popis byl vytvořen automaticky

a) neorientovaná hrana,

b) přímá orientovaná hrana,

c) a d) násobné hrany,

e) a f) rovnoběžné hrany,

g) orientovaná smyčka,

h) neorientovaná smyčka,

i) a j) násobné hrany se smyčkou

Matice sousednosti:

Využívá se u:

* grafů s velkým počtem hran a
* grafů, které se v průběhu algoritmu často mění.

Matice sousednosti grafu 𝐺 je čtvercová matice 𝐴 = (𝑎𝑖𝑗), řádu 𝑛, ve které:

* prvek (𝑎𝑖𝑗) = 1 právě tehdy, když jsou vrcholy 𝑖 a 𝑗 sousední.
* prvek (𝑎𝑖𝑗) = 0 právě tehdy, když jsou vrcholy 𝑖 a 𝑗 nesousední

Obsah obrázku diagram

Popis byl vytvořen automaticky

Matice incidence:

Využívá se u:

* Grafů s menším počtem hran a
* Grafů, u kterých se v průběhu algoritmu nemění počet hran.

Incidenční matice 𝐵 grafu 𝐺 je matice obdélníková s 𝑛 řádky a 𝑚 = |𝐸(𝐺)| sloupci.

* Pro ∀
  + vrchol grafu 𝐺 platí, že mu odpovídá jeden řádek matice 𝐵
  + hranu grafu 𝐺 platí, že ji odpovídá jeden sloupec matice 𝐵.

Formální definice:

Nechť 𝐺 = (𝑉, 𝐸, 𝜌) je neorientovaný graf, kde

množina vrcholů je 𝑉 = {𝑣1, 𝑣2, … , 𝑣𝑛},

množina hran je 𝐸 = {𝑒1,𝑒2, … ,𝑒𝑚} a

incidence je 𝜌: 𝐸 ⟶ {[𝑉1,𝑉2]}, kde 𝑉1,𝑉2 ∈ 𝑉 přiřazuje každé hraně 𝐸 grafu 𝐺 neprázdnou množinu dvojic uzlů z množiny 𝑉. (lidsky: 𝜌 označuje dvojici vrcholů, které jsou danou hranou spojeny.)

Maticí incidence grafu 𝐺 nazýváme obdélníkovou matici 𝐵 = (𝑏𝑖𝑗), ve které:

* prvek (𝑏𝑖𝑗) = 1 právě tehdy, když hrana 𝑒𝑘 inciduje s uzlem 𝑣𝑖
* prvek (𝑏𝑖𝑗) = 0 v opačném případě.

Obsah obrázku tabulka

Popis byl vytvořen automaticky

1. **Úplný graf**

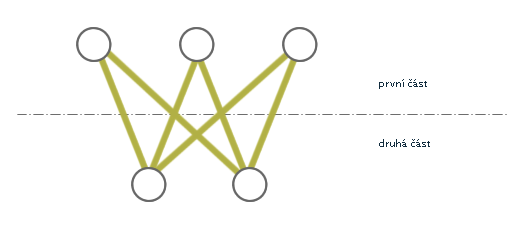
úplný graf se dá reprezentovat třídou ve škole, kde každý každého zná. Všechny vrcholy grafu jsou propojeny navzájem

Obsah obrázku interiér, světlo, ve tmě, rozostření

Popis byl vytvořen automaticky

1. **Bipartitní graf**

Bipartitní graf vznikne rozdělením grafu na dvě části. Jedna strana může být vždy spojena jen s druhou stranou



1. **Strom**

stromový graf se řídí jedním hlavním pravidlem a to, že se nesmí zacyklit.

Vzužití:

* rodokmen,
* soutěže založené na vylučovacím principu (dál postupuje jeden ze soupeřů),
* botanický klíč
* systém souborů uložených v adresářích, (definovaném rekurzivně) jako množinu souborů a adresářů.Les je graf, jehož komponenty jsou stromy.

Každý potomek který už nemá žádné další potomky se nazývá list.

Strom s 𝒏 vrcholy má právě 𝒏 − 𝟏 hran.

Kořen stromu (root)

* Nejvyšší vrchol (uzel) stromu
* Kořen je jediným vrcholem bez otce
* V každém stromu se nachází nejvýše jeden kořen
* Cesta je posloupnost po sobě jdoucích vrcholů, které jsou spojeny jednou, nebo více

hranami.

* Délka cesty je počet hran cesty. Délku cesty z vrcholu k sobě samému potom můžeme definovat jako nulovou. Ke každému vrcholu je z kořene právě jedna cesta.
* Hloubka vrcholu ve stromě (depth) (úroveň, na které se vrchol nachází) je definována jako délka této cesty. Úroveň (hloubka) kořene stromu je tedy nulov
* Výška stromu je maximální hloubka vrcholu stromu.
* Šířkou stromu rozumíme počet uzlů na stejné úrovni.

Obsah obrázku diagram

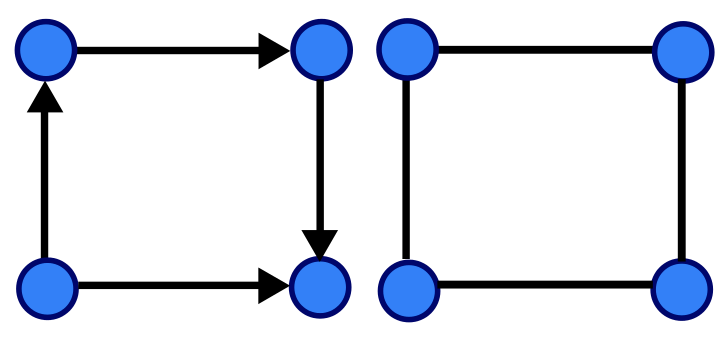
Popis byl vytvořen automaticky

1. **Orientovaný graf**

v reálných situacích často nastává možnost, že je možné postupovat pouze jedním směrem.

V takovém případě se jedná o orientovaný graf. Nejbližší reprezentací z reálného života by mohla být jednosměrná silnice

*na obrázku orientovaný graf (levo) a neorientovaný graf (pravo)*



Pro některé účely je zapotřebí vrcholy grafu **ohodnotit**. Hodnocení většinou vyjadřuje reálné číslo

# Prohledávání stavového prostoru

Algoritmy pro prohledávání stavového prostoru se používají tehdy, když není možné získat výsledek přímým výpočtem, nebo by výpočet byl neúměrně složitý

ideálním příkladem může být šachová hra, hlavolam Hanojské věže a Loydova patnáctka

Stavový prostor lze reprezentovat orientovaným grafem 𝑮 = (𝑽,𝑬). Graf 𝑮 je stromem.

* + Uzly 𝑉 reprezentují stav
  + Hrany 𝐸 reprezentují přechody mezi stavy.

*Metody prohledávání lze rozdělit do 3 základních skupin:*

*1. Slepé*

* *úplné prohledávání nevyužívající žádné dodatečné informace*
* *zde postupně aplikujeme všechny použitelné operátory*
* *příklad: BFS, DFS, uniform-cost, iterativní prohlubování, …*

*2. Heuristické*

* *úplné nebo částečné prohledávání využívající hodnocení zvolené cesty*
* *Zde operátory vybíráme na základě daného kritéria*
* *příklad: beam-search, hill-climbing, best-first search, algoritmus A\*,*

*3. Náhodné*

* + *volíme operátory náhodně*

Existují 3 základní vlastnosti, podle kterých lze metody hodnotit:

1. Časová složitost

* + Minimální/maximální/průměrný čas potřebný k vyřešení úlohy danou metodou
  + Čas je přímo závislý na výpočetním výkonu
  + V praxi se často reprezentuje např. jako počet prozkoumaných stavů v daném čase (objektivnější hodnota)

2. Prostorová složitost

* + Minimální/maximální/průměrné množství operační paměti, potřebné k vyřešení úlohy danou metodou
  + Nezávislost na platformě ⇒ náhrada údaje o počtu MB → počet stavů současně uchovaných v paměti

3. Kvalita získaných výsledků

* + Hodnota, která udává výpověď o tom, zda je daná metoda:

úplná (tj. vždy, když existuje řešení, tak jej nalezne),

optimální (tj. nalezené řešení je nejlepší ze všech), apod.

Slepé metody:

* Neinformované metody
* Nemají k dispozici žádné vhodné znalosti o stavovém prostoru, které by

mohly urychlit cestu k cíli

* Systematické procházení všech uzlů

Prohledávání do šířky (BFS)

= breadth-first search

* Následníci vybíráni pro expanzi ze seznamu typu fronta
* Postupně se prochází strom řešení po vrstvách (prohledáme veškeré uzly, které mají nižší hloubku, než je hloubka koncového stavu)
* Každý uzel je navštíven nejvýše jednou.
* Vždy je nalezeno optimální řešení (koncový stav s nemenší hloubkou)
* Obsah obrázku diagram

  Popis byl vytvořen automaticky

Prohledávání do hloubky (DFS)

= depth-first search

* Následníci jsou vybíráni vybírání pro expanzi ze seznamu typu zásobník
* BFS, zde můžeme některými uzly procházet vícekrát (často se musíme vracet (tzv. backtracking))
* Nemusíme nalézt optimální řešení (koncový stav s nejmenší hloubkou) či dokonce žádné řešení (pokud má stavový prostor nekonečnou větev)
* V úlohách s konečným stavovým prostorem a s jedním koncovým stavem nalezneme stejné řešení jako u BFS
* Obsah obrázku diagram

  Popis byl vytvořen automaticky

Heuristické metody prohledávání stavového prostoru

= Informované metody

* Mají znalosti o stavovém prostoru (umožňují jim odhadnout délku cesty koncového stavu od aktuálního)
* Odhad reprezentuje tzv. heuristická funkce ℎ(𝑛): Čím nižší hodnoty ℎ(𝑛) nabývá, tím je vyšší pravděpodobnost, že cesta k řešení povede skrze stav 𝑛
* Heuristickou funkci dodává člověk (na základě znalostí), informované metody jsou na heuristické funkci kriticky závislé
* Čím je heuristika (kritérium) lepší, tím rychleji a s menším zatížením paměti dojde k nalezení řešení.
* snažíme přidat nějakou přidanou hodnotu (logiku) nemusím procházet všechny stavy protože se můžu dostat do stavu kde už to logicky není potřeba

# 1 Stavový prostor. 1.1 Základní definice - PDF Stažení zdarma

stavový prostor je prostor všech stavů které můžou nastat

# Řadící algoritmy

Řadící algoritmus je algoritmus zajišťující uspořádání dané sady (typicky pole) do požadovaného pořadí

* nejvýkonnější algoritmy bývají zpravidla ty, které neporovnávají jednotlivé hodnoty prvků, ale fungují na jiném principu
* existují algoritmy stabilní a nestabilní. U stabilního algoritmu nehrozí, že by v jeho průběhu byly prohozeny stejné hodnoty
* to je užitečné především tehdy, když je řazeno více algoritmů

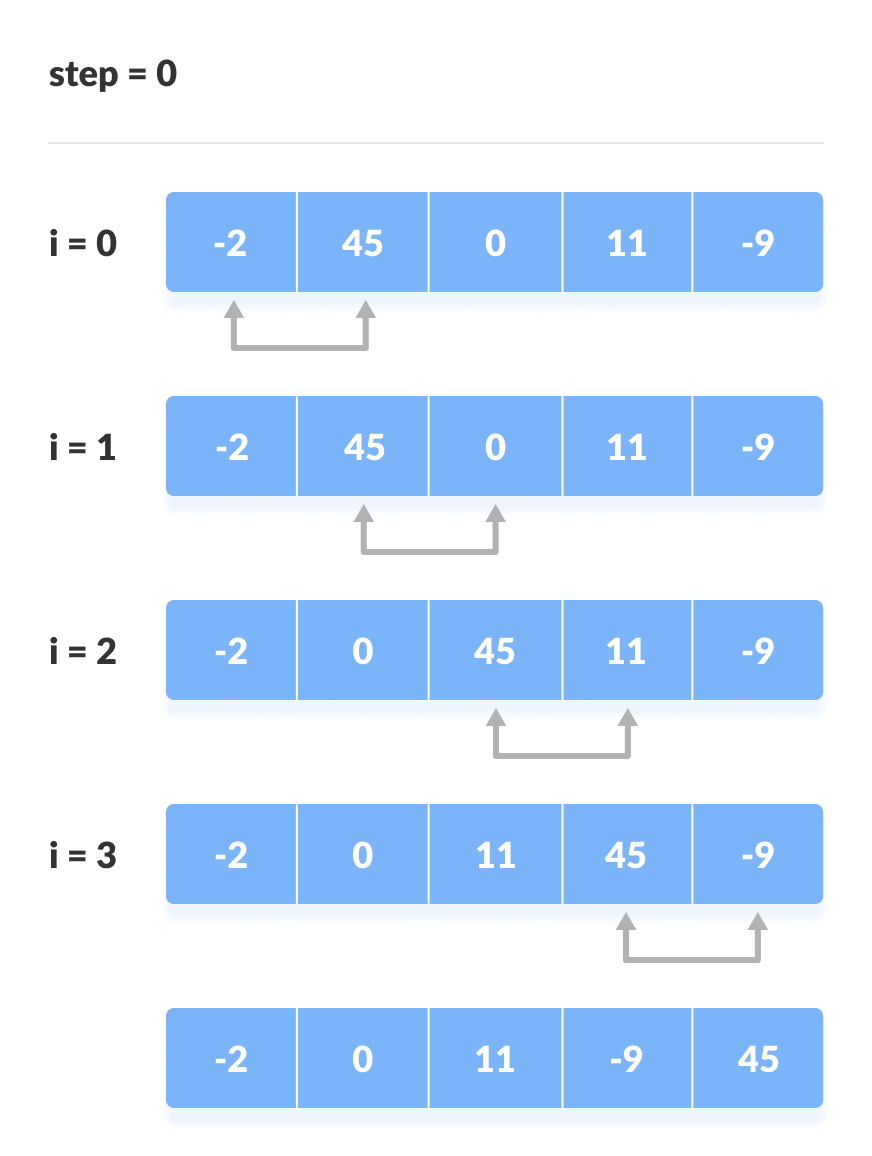
Typy sortů

Bubble sort

Bubble je stabilní algoritmus. Porovná vždy 2 sousední prvky. Pokud je nižší prvek nalevo od vyššího, prohodí je a pokračuje k dalšímu indexu, kde aplikuje stejnou logiku.

Pokud jsou čísla správně (nižší napravo od vyššího), pokračuje k dalšímu indexu bez úpravy

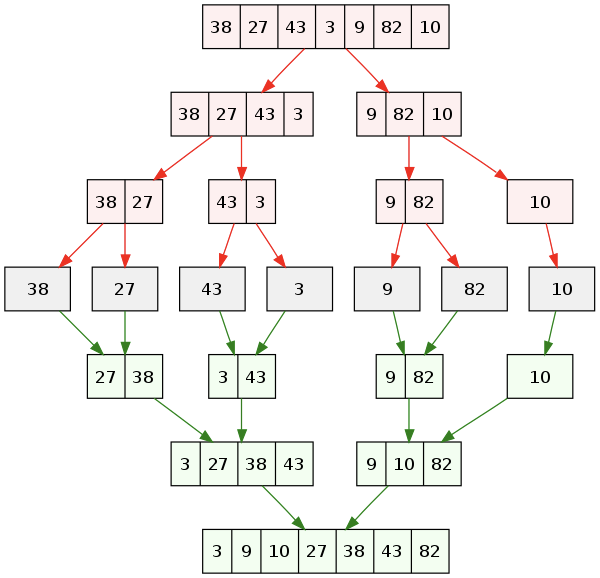
Tento zmíněný algoritmus zajistí, že na konci bude vždy nejnižší hodnota. Průběh bubblesortu je vyobrazen na následujícím obrázku



Merge sort

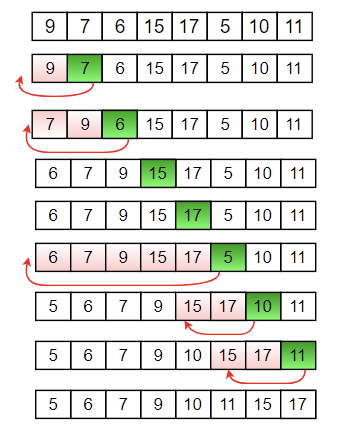
Merge sort, někdy také nazýván metodou slévání, je stabilním algoritmem pracující na principu slévání polí. Tento algoritmus je nejsnazší pochopit na samotném výpočtu, jeho prezentace může být někdy poněkud složitá. (viz. obrázek)

Původní pole je uprostřed rozděleno na dvě stejně dlouhé části. Pokud se jedná o pole o lichém počtu prvků, pole na levé straně musí vždy být o větším počtu prvků. Tyto dvě vzniklé pole jsou dále děleny, dokud nejsou rozděleny na jednotlivé prvky. V moment, kdy se tak stane, začne samotný princip slévání. Sléváme tak, že porovnáme prvky vedlejších polí. Větší prvek je v ten moment sepsán do následujícího pole. Pokud jsou prvky již jen dva, menší prvek ho následuje. V dalším kroku sléváme již pole o více prvcích, tudíž musíme nasadit odpovídající strategii. Vezmeme prvek nejvíc vlevo ( Vzhledem k seřazení v minulém kroku je to ten větší) a porovnáme ho s prvním (největším prvkem) v druhém poli. Větší prvek (budeme počítat s tím, že by to byl ten z levého pole) je sepsán a odstraněn z pole. Následuje porovnání druhého prvku v levém poli (menší číslo, to co zbylo) s prvním prvkem (větším) v pravém poli. Větší prvek je opět sepsán do dalšího pole a smazán z originálního pole. Tímto způsobem zlikvidujeme všechny prvky v poli a postoupíme na další. Hotovi jsme v moment, kdy opět zbývá jen jedno pole.

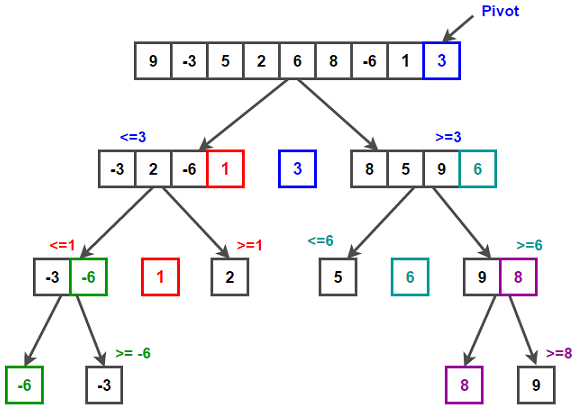


Insertion sort

Insertion sort, někdy také metoda vkládáním je nestabilním algoritmem. Hned zpočátku si řekneme, že prvek nejvíce nalevo je triviálně seřazen. Vezmeme následující prvek a řadíme ho v poměru s triviálně zařazeným prvním prvkem na příslušnou pozici (pokud řadíme směrem k vyšším číslům a hodnota prvku je nižší než hodnota toho triviálně seřazeného, umístíme ho nalevo od něj. Pokud je vyšší, napravo). V ten moment máme seřazená dva prvky a princip se opakuje, dokud není celé pole seřazeno.



Quicksort



Základní myšlenkou algoritmu rychlého řazení je rozdělení řazené posloupnosti čísel na dvě přibližně stejné části (rychlé řazení patří mezi algoritmy typu rozděl a panuj). V jedné části jsou čísla větší a ve druhé menší, než nějaká zvolená hodnota (nazývaná pivot – anglicky „střed otáčení“). Pokud je tato hodnota zvolena dobře, jsou obě části přibližně stejně velké. Pokud budou obě části samostatně seřazeny, je seřazené i celé pole. Obě části se pak rekurzivně řadí stejným postupem, což ale neznamená, že implementace musí taky použít rekurzi.

Zde je seznam některých metod:

* První prvek – popřípadě kterákoli jiná pevná pozice. (Pevná volba prvního prvku je velmi nevýhodná na částečně seřazených množinách.)
* Náhodný prvek – často používaná metoda. Průměr přes každá data je O(N log N), přičemž zde se průměr bere přes všechny možné volby pivotů (rozděleno rovnoměrně). Nejhorší případ zůstává O(N2), protože pro každá data může náhoda nebo Velmi Inteligentní Protivník vybírat soustavně nevhodného pivota, např. druhé největší číslo. V praxi většinou není dostupný generátor skutečně náhodných čísel, proto se používá pseudonáhodný výběr.
* Metoda mediánu tří – případně pěti či jiného počtu prvků. Pomocí pseudonáhodného algoritmu (také se používají fixní pozice, typicky první, prostřední a poslední) se vybere několik prvků z množiny, ze kterých se vybere medián, a ten je použit jako pivot.