Algoritmizace - Grafy, Prohledávání stavového prostoru, Řazení

*předmluva: Na každém z daných témat se dá strávit mnoho času, ideálním postupem je naučit se prezentaci tak, jak je uvedena v dokumentu chronologicky. Moje doporučení je podrobně mluvit o Hanojských věžích a využívat podobností s šachovou hrou, neboť ne každý z nás je dobrým programátorem a princip stolní hry by mu mohl sednout více.*

*Andreas Dumalas*

# 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

neexistence přímého výpočtu neznamená, že nemůžeme vytvořit strategii, která řešení nalezne

zavedeme si označení pro stavový prostor SP = (S,A) - S je stav, kterého úloha nabyla, A je množinou přípustných operátorů pro přechod mezi stavy

k-tého stavu (Sk) (S s indexem k) dosáhneme tím, že aplikujeme operátor přechodu A(ik) (A s indexem přechodu ze stavu i na stav k) (i v abecedě před písmenem k, tudíž reprezentuje následující stav)

Kódování stavového prostoru

Příklad kódování stavového prostoru si předvedeme na příkladu Hanojských věží.

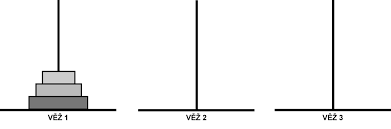
problém Hanojských věží spočívá v tom, že máme tři věže a tři disky, jenž jsou umístěny od největší po nejmenší (viz obrázek)

*pracujeme se třemi pravidly:*

1) tah znamená uchopit jeden kotouč a umístit ho na jinou věž

2) během jednoho tahu lze vždy přemístit jen a pouze jeden kotouč a to ten, který je na vrchu.

3) je zakázáno umístit větší kotouč na menší



Stavem S0 = (111) (S s indexem 0) rozumíme počáteční stav, ve kterém jsou veškeré kotouče na první věži seřazené od nejmenší po největší

-v trojčíslí (111) pozice čísla definuje, o který kotouč se jedná

-v trojčíslí (111) hodnota čísla definuje, na které věži se kotouč nachází

-pokud by vrchní kotouč z první věže byl umístěn na třetí, vzniklo by trojčíslí (113)

-pokud by následně druhý (prostření) kotouč byl umístěn na druhou věž, vznikne (123)

-pokud by byl nejmenší kotouč byl umístěn zpět na první věž, vzniklo by (121)

-*cílem hry je dostat veškeré kotouče ve stejném pořadí na druhou věž, tudíž uvést je do*

*cílového stavu g = (222) (g = cílový stav)*

Počet aplikací operátorů nazýváme délkou cesty. Nalezenou posloupnost operátorů vedoucí k požadovanému výsledku nazýváme cestou z počátečního do cílového stavu

*Zavedeme si kroky pro vyřešení úlohy stavového prostoru:*

1. Zavedeme kódování stavů ( pro nás (111))
2. Identifikujeme operátory (pro nás je operátor přenos disku)
3. Identifikujeme cíle ( pro nás (222), alias cílový stav)
4. zvolíme strategii (o strategiích si povíme za chvilku)
5. Aplikujeme strategii na počáteční stav (111)
6. V každém kroku proběhne kontrola, jestli se jedná o cílový stav (222)

Metody prohledávání stavového prostoru

*Stavový prostor může být velmi rozsáhlý, nebo dokonce nekonečný. Pro tento případ si zavádíme metody (strategie) prohledávání stavového prostoru*

1. Metoda neinformovaná
2. Metoda informovaná
3. Metoda lokální

**Metoda neinformovaná** pracuje na principu slepce v labyrintu. Volí operátory slepě, nemá informaci o výhodnosti tahu. Nehodnotí cestu a neodhaduje, zdali se blíží cíli. Neinformovaná metoda je nenáročná na paměť a snadná na implementaci

**Metoda informovaná** se dá přirovnat šachistovi v turnaji. Odhaduje, zdali bude tah výhodný na základě analýzy předchozích tahů a momentální situace. Na základě této situace se snaží vyhodnotit situaci na několik tahů dopředu a pak dle svého odhadu a intuice volí tah. Tomuto letmému volení tahu říkáme heuristika. Ve hře šachy hráč s lepší heuristikou zpravidla vyhrává

**Metoda lokální** se dá popsat jako šachista, který přijde o paměť pokaždé, co odehraje svůj tah. Snaží se stále vyhodnotit výhodnost tahu a zdali se blíží k cíli, jen se nerozhoduje na základě paměti a vědomí o minulých stavech, jak by to informovaná metoda udělala. Pro lokální metodu ale hrozí zacyklení. Šachista který udělal v jednom bodě tah věží o čtyři políčka dopředu, může po ztrátě paměti vyhodnotit, že nejlepší možný tah je o čtyři políčka dozadu. To by znamenalo, že po uplynutí dvou jeho tahů se nachází stále ve stejné pozici.

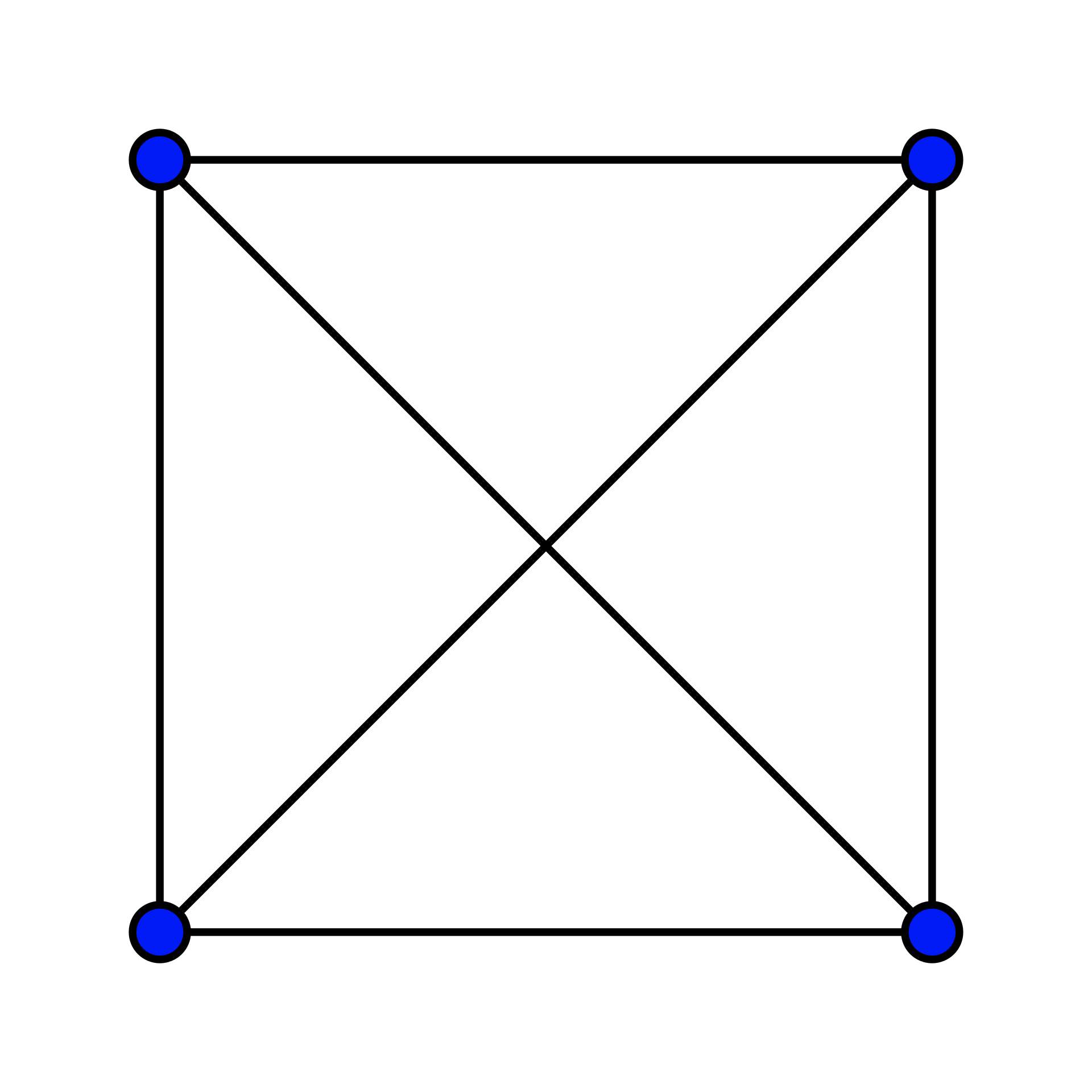
# Teorie grafů

historie - grafy vznikly původně mimo obor informačních technologií a počítačů, když v 18. století Leonhard Euler (je po něm pojmenovano Eulerovo číslo) musel vyřešit matematický problém, když musel překonat sedm mostů a vrátit se na původní místo, aniž by použil jeden most dvakrát. Použil k tomu něco, čemu dnes říkáme orientovaný graf.

Základní typy grafů:

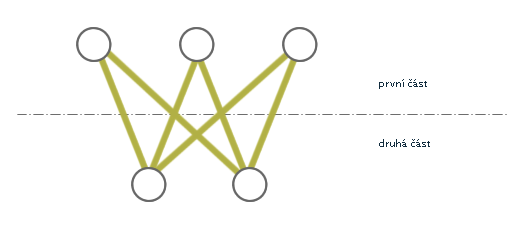
1. **Úplný graf**

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



1. **Bipartitní graf**

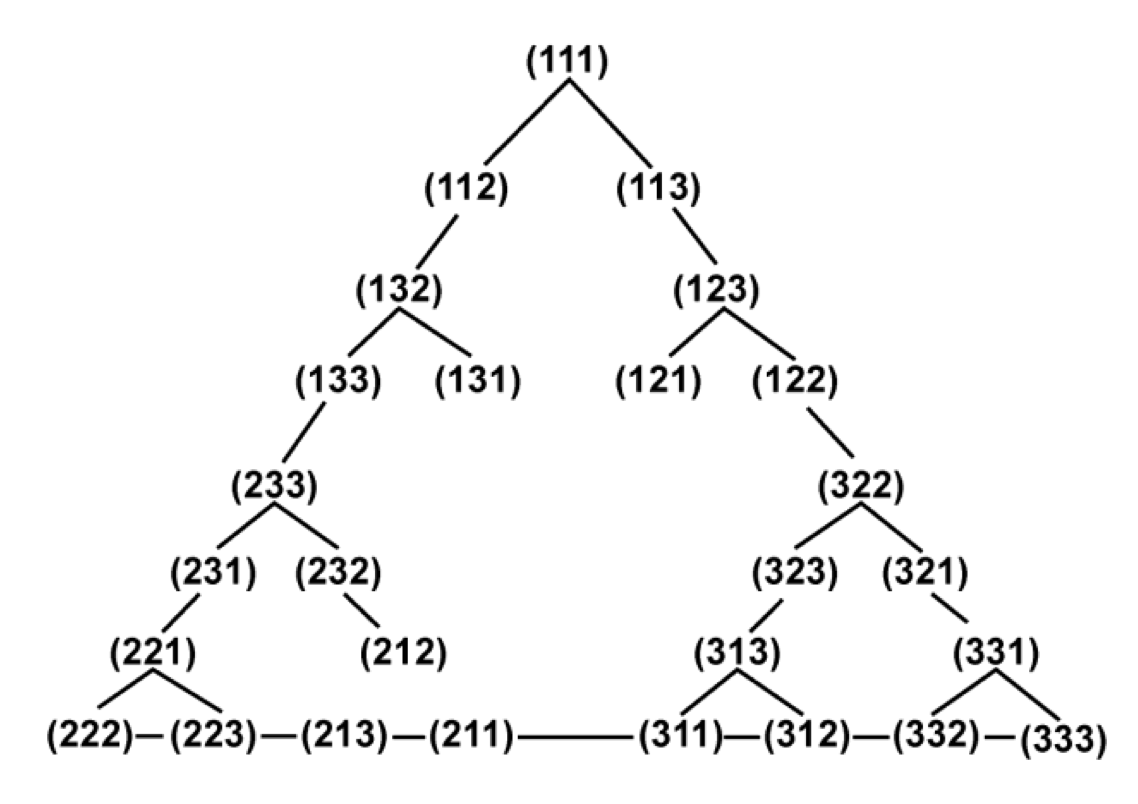
Bipartitní graf vznikne rozdělením grafu na dvě části. Jedna strana může být vždy spojena jen s druhou stranou



1. **Strom**

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

Ideálním využitím Stromového grafu je například kódování problému Hanojských věží

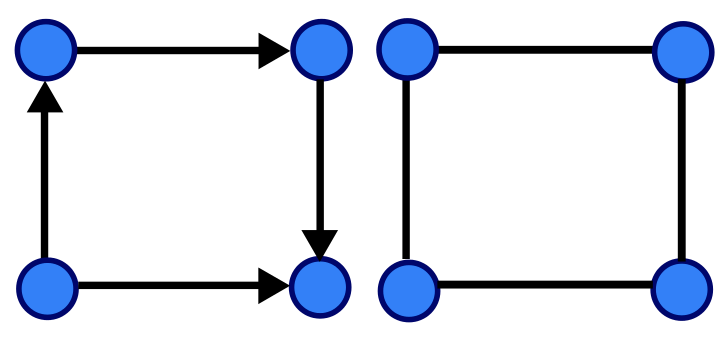


1. **Orientovaný graf**

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

# 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

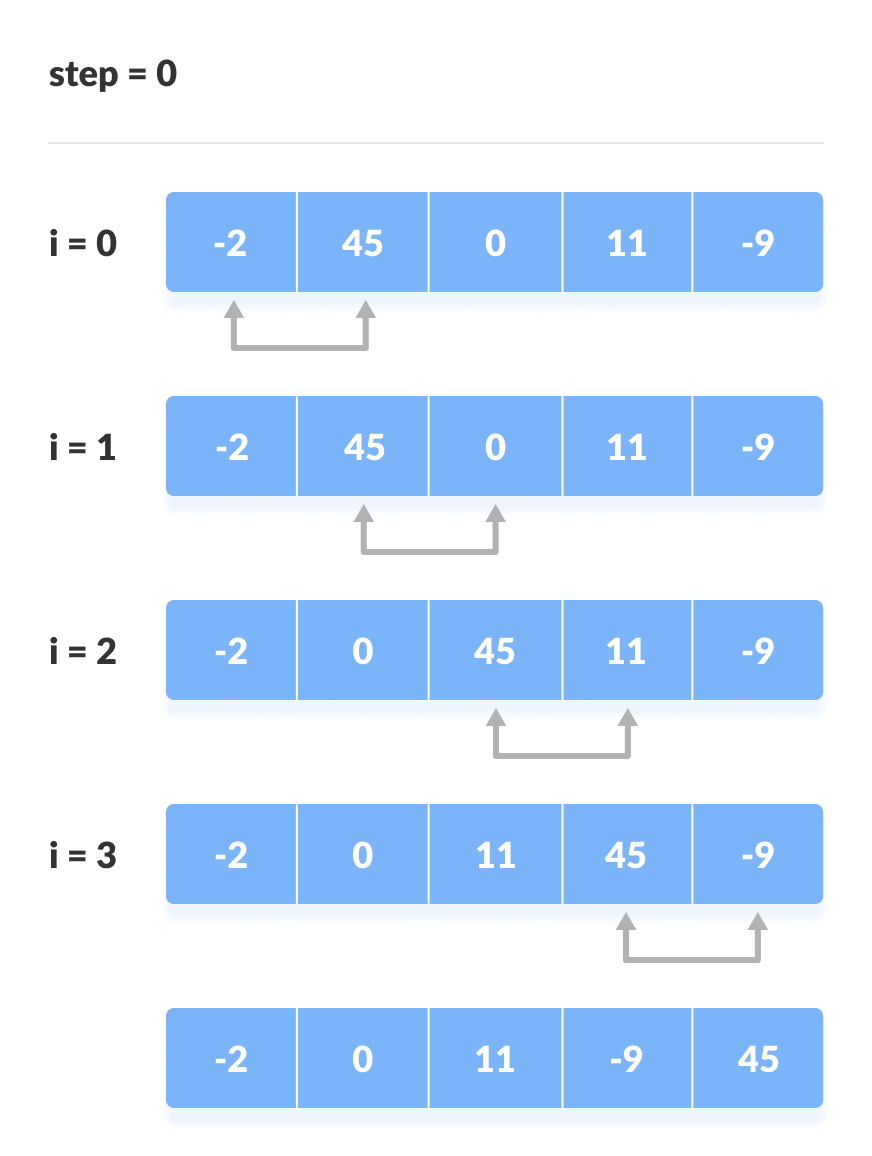
Typy sortů

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

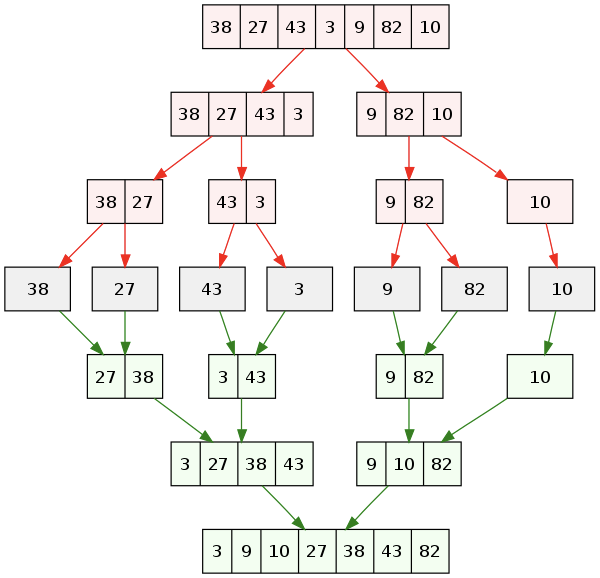
Tento zmíněný algoritmus zajistí, že na konci bude vždy nejnižší hodnota. Průběh bubblesortu je vyobrazen na následujícím obrázku



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.

