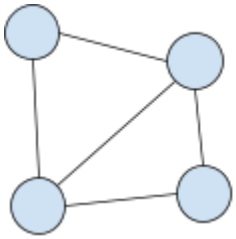
2. Algoritmizace - Grafy, Prohlédávání stavového prostoru, Řazení

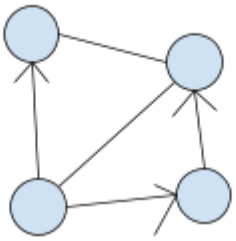
graf - způsob reprezentace vztahu mezi daty, je tvoreny vrcholy = objekty a hranami = spojeni dvou vrcholu

graf x strom = strom je podmnozina grafu

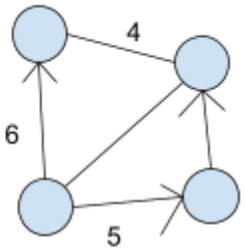
orientovany graf - usporadana dvojice, maj proste nejak urcenej smer sipky proste
neorinetovany graf - mnozina prvku, nemaji urceny smer bez sipek, proste jenom spojeny vrcholy bez zadnyho smeru
ohodnoceny graf - graf rika jaka je vzdalenost mezi urcitymi vrcholy, delka hrany proste,
uplny graf - kazdej vrchol je spojenej s kazdym pomoci hrany



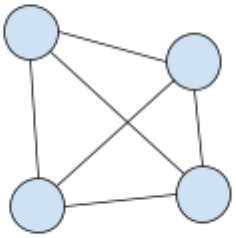
Neorientovany



Orientovany



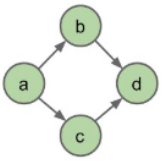
Ohodnoceny



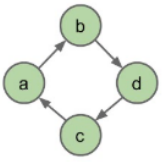
Uplny

cyklicky grafy - muzu chodit dokola nekonecne skrz vsechy body
necyklicky grafy - muzeme se zaseknout a neprojit vsechny body jednou cestou bez vraceni

Acyclic:

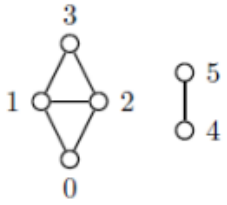


Cyclic:



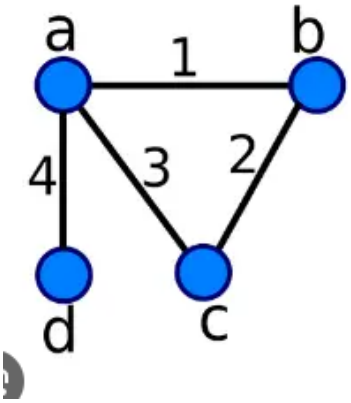
smycka (kruznice) - cyklus mezi vrcholama

matice sousednost -



$$A = \begin{array}{c} v\backslash v \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

matice incidence -



	1	2	3	4
a	1	0	1	1
b	1	1	0	0
c	0	1	1	0
d	0	0	0	1

eulerovksy grafy - vpodstate muzu nactrnout ten graf jednim tahem, kazdy vrchol muze bejt prochazenej jenom jednou

Statovy prostor
stavovy prostor je soubor vsech stavu, ktere mohou v urcitem problemu nastat. Je mozne ho ulozit do ruznych datovych struktur, nejvyhodnejsi je moznost stromu

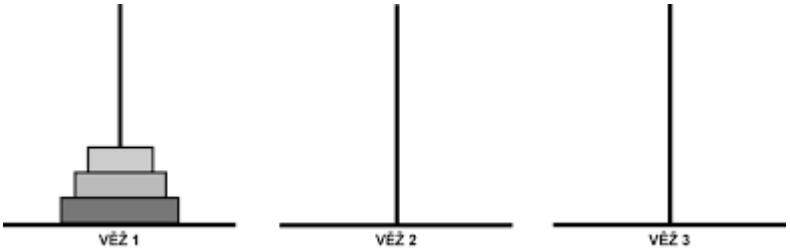
Prohledavani stavoveho programu je fukce, ve kterem se snazime najit co nejoptimalnejsi stav ktery splnuje to co chceme.

Prohledavat ho muzeme - vse se prochazi brute forcem

1. Do sirky (BFS) - prohledava stavovy prostor vrstvu po vrstve, zacina v pocatecnim vrcholu a navstvi jeho sousedy, potom dalsi, potom dalsi….
2. Do hloubky (DFS) - chodi po vetvich po jedny vetvi, potom zkousi dalsi,dalsi…

Moznost prochazeni taky heuristickyma algoritmama - pouziva logiku

Prikladem jsou napriklad hanojske veze - mame tri vezicky s tremi disky, ktere jsou umisteny od nejvetsiho po nejmensiho, my se snazime je premistit tak,ze nemuzeme skladat vetsi disk na ten mensi a snazime se celou vez postavit na 3 vezi tak aby jsme dodrzovali pravidla



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:

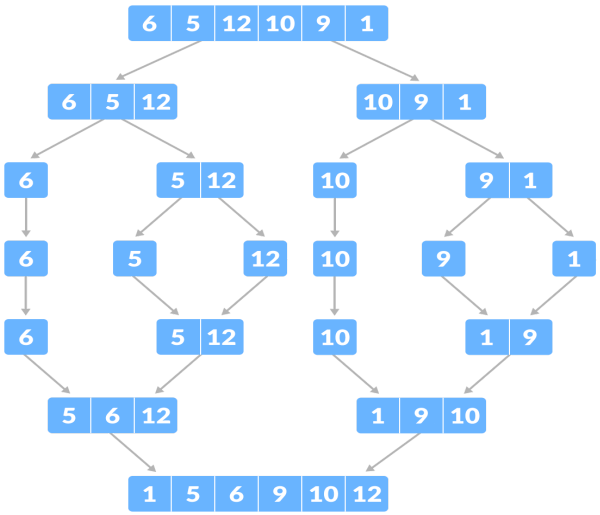
1)
Zavedeme kódování stavů (pro nás (111))
2)
Identifikujeme operátory (pro nás je operátor přenos disku)

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)

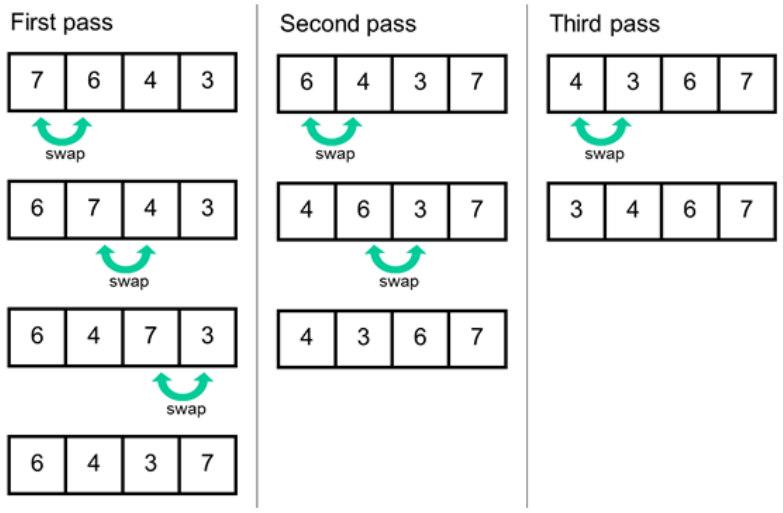
Razeni
operace, behem ktere srovnavame urcitou datovou strukturu, ktera je naplnena daty. Existuji ruzne typy radicich algoritmu - kazdy je vhodny na urcity dataset, delka, pocet dat atd. Vetsinou proste razeni od nejmensiho po nejvetsiho atd

Druhy radidich algoritmu - Radix, Selection Sort, **Merge Sort**, **Bubble Sort**, Heap Sort, Stalin Sort, Sleep sort, **Quick Sort**....

Merge sort
rekurznivni algoritmus - rozdel a panuj
Algoritmus vezme nerozdelene pole a rozdeli si ho na dve poloviny a deli je na poloviny tak dlouho do ty doby nez zbydou jenom dvojice. Dvojice se nasledne seradi a rekurzivne se vracemi zpatky nahoru. [n2 - kvadraticka]



Bubble sort
Algortimus vezme nerozdelene pole a kazde dva sousedicic prvky porovna a usporada je podle velikosti, postupnou iteraci dojde k usporadani. Potom co je to usporadany tak dobre napsanej algoritmus tak kontrolujeme jestli neproslo k prohozeni mame hotovo. Hodis to proste jako podminku. [n log n]



Quick sort
rekurznivni algoritmus
Na vybrani pivota je vice zpusobu - 1. cislo, prostredni a nebo konecny nebo klido random. Ja si vezmu ten kterej vezme to konecny cislo jako pivot. Porovnavam hodnoty s pivotem a delim je na ty ktery jsou vetsi nez pivot a na cisla ktery jsou mensi nez pivot. Potom vybiram dalsi pivoty a pokracuju do ty doby nez nemam single hodnoty