

## **Abstraktní datový typ - ADT, abstraktní datová struktura - ADS, aplikace paradigmat OOP v datových strukturách**

*Datová struktura* - potřeba zpracování údajů vede k modelování reality, *vymezení objektu zkoumání; abstrakce reality*; DS představují způsob uchování a organizace dat s cílem umožňovat zpřístupňování a modifikaci.

*ADT* je matematická struktura složená ze dvou částí - třídy a operace nad prvky. ADT zapouzdřuje před uživatelem svoji vnitřní strukturu, zajímají nás operace a jejich výsledky, neimplementujeme operace.

*ADS* je konkrétní realizace ADT, konkrétně implementujeme algoritmy operací, uživatel zná jen vstup a očekává výstup.

*Doména* - sestava blíže nespecifikovaných prvků.

*Kategorizace operací ADS/ADT:*

- konstruktory - konstrukce a inicializace struktury
- destruktory - zrušení celé struktury
- selektory - zpřístupňovací operace bez reorganizace struktury
- prohlídky celé struktury

*Klasifikační kritéria DS:*

Logická úroveň - přímý/sekvenční přístup k prvkům, ne/lineární množina - prvky ne/porovnatelné

Fyzická implementační úroveň - elementární, strukturované datové typy

Složení struktury - homogenní, nehomogenní prvky

*Zapouzdění* - neznáme vnitřní uspořádání DS, jen metody, zajímá nás, co nám DS nabízí, ne, jak to dělá

*Dílnost* - možnost vytvoření hierarchie od ADS po uživatelskou DS s konkrétním typem dat. Objekt ADS nevytváříme, jen využíváme jeho dědičných vlastností (vložit, odebrat, prohlídka)

*Polymorfismus* - akce uskutečňované nad různými uživatelskými daty jsou pojmenovány stejně, ale realizují se jinak.

## Metody porovnávání datových struktur - výpočetní složitosti algoritmů a složitosti paměťových reprezentací

Pro vyhodnocení různých algoritmů je nutné vymezit objektivní kritéria, jež umožní provádět jejich vzájemné porovnání

*časová paměťová kritérium* - čas potřebný k provedení algoritmu, paměťový prostor potřebný k realizaci, výpočetní složitost - počet operací, které je nutné vykonat. Důležitost efektivního algoritmu narůstá s přibývajícím množstvím dat.

Velké množství dat - je nutné spolupracovat s externí pamětí - neefektivní swapping (skoková změna složitosti)

Složitost - polynomiální, exponenciální  $O(n^2)$ , logaritmická  $O(\log n)$  - třídění půlením pole, lineární, lineárně-iterativní  $O(n \log_2 n)$  - quicksort, konstantní  $O(1)$

Paměťová složitost je vyjádřena množstvím paměti počítače, jež je nutná pro kód a data. Paměťová velikost kódu je obvykle zanedbatelná proti velikosti dat. Nejčastěji bývá složitost konstantní, lineární nebo logaritmická. O programu, který nevyžaduje více paměťového prostoru než zabírají zpracovávaná data, říkáme, že pracuje in-situ (na svém vlastním prostoru).

## Lineární ADT (pole, seznam, zásobník, fronta) - základní charakteristiky, kategorizace a aplikace

Lineárně uspořádaná množina - existuje-li v množině  $M$  relace  $R$ -lineární, pak je množina  $M$  lineárně uspořádaná. Nejobecnější interpretace lineárně uspořádané množiny může být specifikována jako Super ADT Sekvence. *Sekvence* vymezuje "maximální možnosti":

- zpřístupnění logických vstupních bran
- zpřístupnění nejbližších sousedů
- zpřístupnění prvku s definovaným pořadím od začátku

začátek, konec, následník, předchůdce, bezúrovňová struktura, porovnatelné prvky

*Pole*

- všechny prvky pole existují od okamžiku jeho vytvoření až do okamžiku jeho zrušení
- prvky pole jsou zpřístupňovány na základě svého pořadí od definovaného začátku pole
- klasická operace odeber absentuje

Pro implementaci homogenního pole je vhodné využít organizaci s implicitními vztahy - zpřístupnění  $i$ -tého prvku založeného na výpočtu jeho paměťové adresy při znalosti počáteční adresy pole a paměťové velikosti prvku.

Datové struktury

### Lineární seznamy

Jednosměrně a obousměrně zřetězené. Zpřístupňování prvků je založeno na sekvenčním přístupu.

Implementace - dynamický lineární seznam (nejčastěji), seznam na poli, seznam s hlavou, bez hlavy, ne/cyklicky zřetězené.

U všech typů implementací jsou složitosti základních operací  $O(1)$ , kromě operace prohlídka  $O(n)$ .

Příklad: zastávky v rámci MHD, souprava železničních vagónů

### Zásobník (LIFO)

Standardní typ charakterizovaný zejména možností zpřístupňování pouze jednoho prvku, nacházejícího se na začátku struktury. Vrchol struktury je jediným možným místem struktury pro realizace vkládání a odebírání prvků. Absentuje tradiční operace prohlídka.

Implementace - dynamický zásobník (lineární seznam), zásobník na poli.

Složitost všech operací je  $O(1)$ .

Příklad: zásobník pistole.

### Fronta (FIFO)

Standardní typ disponující jednu vstupní, resp. výstupní bránou, které umožňují vkládání prvků do, resp. odebírání ze struktury.

Implementace - dynamická fronta (obousměrný lineární seznam), fronta na poli

Složitost základní operací je  $O(1)$ .

Příklad: fronta klientů

## Hierarchické ADT (unární strom, binární strom, k-cestný strom) - základní charakteristiky, kategorizace a aplikace

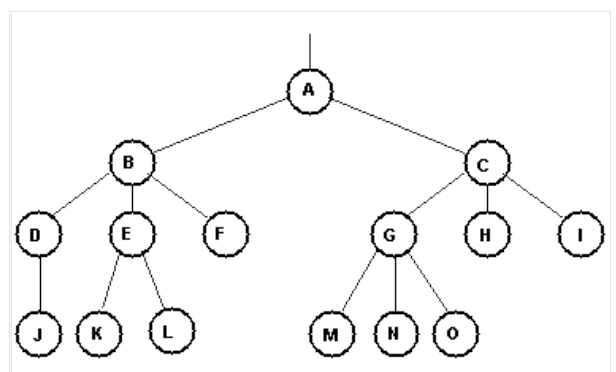
Existuje v množině  $M$  relace  $R$  - hierarchické uspořádání, pak o množině  $M$  říkáme, že je hierarchicky uspořádaná.

kořen, listy, potomek/syn, předek/otec, úroňová struktura, neporovnatelné prvky

*Unární strom* - každý prvek unárního stromu má přístup pouze k jednomu prvku (otci)

Implementace: kořen  $a_1$ , listy:  $b_2, d_4, j_{10}, \dots$

Příklad: vodstvo, kanalizační systém



*Binární strom* - každý prvek binárního stromu má maximálně dva syny, strom je uspořádaný - rozlišení pravého a levého syna.

Implementace - *dynamický binární strom* - organizace s explicitními vztahy, orientovaný směrem od kořene k listům, zpřístupní bratra a otce nejsou základní operace; *binární strom na poli* - organizace s implicitními vztahy, implementujícím typem je homogenní souvislé pole

Složitosti - složitosti základních operací  $O(1)$ , kromě operace prohlídka

Prohlídka: PreOrder akce-levá-pravá, InOrder levá-akce-pravá, PostOrder levá-pravá-akce

Příklad: symbolické vyjádření matematických výrazů

*Uspořádaný k-cestný strom*

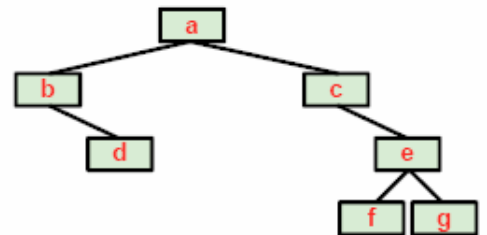
disponuje-li kořenový strom vlastností, že každý z jeho prvků může mít maximálně k synů

Uspořádaný strom je strom, pro nějž platí, že synové každého jeho prvku představují lineárně uspořádanou množinu.

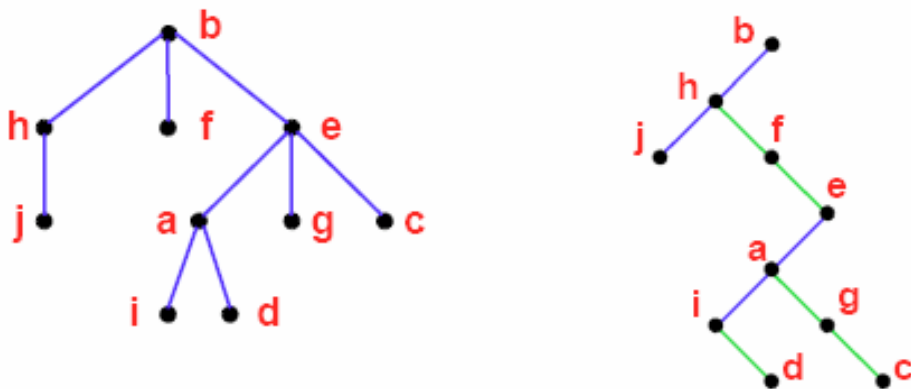
Prohlídky: do hloubky - LIFO, do šířky - FIFO

### Prohlídky

- Preorder (a b d c e f g)
- Inorder (b d a c f e g)
- Postorder (d b f g e c a)



### Princip transformace k-cestného stromu na binární strom



## ADT prioritní fronta - základní charakteristiky, kategorizace a aplikace

Množina s lineárním uspořádáním, přičemž uspořádání je určováno prioritami prvků vzhledem k jejich pořadí odebíráním ze struktury.

Speciální případy prioritní fronty:

Zásobník - LIFO, nejvyšší prioritu má časově nejmladší prvek; Fronta - FIFO, nejvyšší prioritu má časově nejstarší prvek. prioritní fronta nad lineárním seznamem (ne/utříděný)

Datové struktury

Implementace

dvojseznamová struktura - utříděná (časté vyhledávání), neutříděná (časté vkládání)

dvojúrovňová prioritní fronta - bazová struktura - utříděný seznam, přístupová struktura - utříděné vyhledávací pole

Nevýhodou dvojseznamové a dvojúrovňové prioritní fronty je složitost samoorganizujícího mechanismu při nutných reorganizacích.

*Prioritní fronta na haldu* - halda je označení pro binární strom, pro jehož každý vrchol platí, že jeho priorita je větší než priority obou jeho synů (pokud existují). Halda vytváří vždy vyvážený strom - úrovně listů se liší nejvýše o jednotku.

Příklad: fronta kamionů na celní odbavení (kazící se náklad)

## **ADT tabulka - základní charakteristiky, kategorizace a aplikace**

Množina s lineárním uspořádáním, přičemž uspořádání je určováno jednoznačnými klíčovými hodnotami prvků. (Abecední seznam studentů)

Implementace:

tabulka na poli, tabulka na seznamu, tabulka na binární vyhledávacím stromu, tabulka na implicitní kosočtvercové vyhledávací síti, rozptýlené (hashovací) tabulky

algoritmy třídění tabulek: přímým vkládáním (insert sort), výběrem (selection sort), přímou výměnou, výměnou s rozdělováním (quick sort), spojováním (merge sort), číslicové (radix sort)

*Binární vyhledávací strom*

BVS je označení pro binární strom, pro jehož každý vrchol  $x$ , charakterizovaný klíčem  $K(x)$  platí: je-li vrchol  $y$  z levého podstromu vrcholu  $x$ , pak  $K(y) \leq K(x)$  a je-li vrchol  $y$  z pravého podstromu vrcholu  $x$ , pak  $K(y) > K(x)$

Operace Najdi, Vlož, Odeber, Prohlídka - InOrder představuje výstup v utříděném pořadí

Pro každý vrchol dokonale vyváženého binárního stromu platí, že počet vrcholů v jeho levém a pravém podstromu se liší nejvíce o jedna.

Složitost: vlož, odeber, najdi -  $\log_2 n$

*Implicitní kosočtvercová vyhledávací síť*

utříděný ve vzestupném pořadí ve směru logických šipek (zdola šikmo vpravo nahoru, shora šikmo vpravo dolů). Pro každý prvek platí, že jeho následník vpravo dole má větší hodnotu klíče a předchůdce vlevo dole má větší hodnotu klíče a předchůdce vlevo dole menší hodnotu klíče -> prvky na jedné úrovni jsou též utříděny vzestupně doprava

Operace najdi, vlož, odeber mají složitost  $O(\sqrt{n})$

Datové struktury

Rozptýlené tabulky (hashovací tabulky)

Princip hashovacích tabulek spočívá v určení adresy (indexu) hledaného prvku přímo z jeho klíče (z jeho transformace) bez nutnosti uplatnění asociativního algoritmu založeného na porovnávání hledaného klíče s klíči uložených prvků. Transformace klíče je nutná k tomu, aby byl omezen prostor možných hodnot klíče, jenž je daleko větší než prostor použitelných adres pro uložení záznamů

Příklad: identifikace železničních vozů

Implementace pole-seznam

operace vlož - výpočet primární adresy prvku, je-li volná, je vkládaný prvek uložen, pokud je obsazena, je vkládaný prvek uložen do nové, dynamicky vyblokované paměťové místo a je připojen k seznamu prvků této kolizní skupiny

operace najdi/odeber - výpočet primární adresy prvku

Při vhodné volbě rozptylové funkce činí složitosti základních operací Najdi, Vlož, Zruš  $O(1)$

Příklad: kartotéky u lékaře, elektronické slovníky, telefonní seznam

## **ADT graf - základní charakteristiky, kategorizace a aplikace**

Abstraktní datový typ Graf (odrážející binární relaci v množině) představuje heterogenní bipartitní strukturu pracující se dvěma odlišnými třídami prvků - vrcholy a hranami.

*Vrcholov orientovaný p ístup*

Vstupní branou do datové struktury je vrchol(y), s nímž jsou spojeny další informace/data o jeho následnících, resp. předchůdcích a tím i o incidentních hranách. Při vyhledávání ve struktuře jsou přímo k dispozici informace o sousedních vrcholech (následnících a předchůdcích) a tím i informace o incidentních hranách.

Hvězdy mohou být realizovány jako čtyři abstraktní typy: pole-pole, pole-tabulka, tabulka-pole, tabulka-tabulka

Hranově ohodnocený přístup

Vyhledávací operace orientovány na vyhledávání hran - zpřístupnění vrcholů je až druhotné. Hranově orientovaný přístup je vhodný pro aplikace pracující s úseky dopravních sítí, po nichž se přemísťují dopravní prostředky. Označení opačných konců hran pomocí polaritních znamének umožňuje adresovat opačné konce úseků.

K identifikaci hran se používá - dvojice incidentních vrcholů (tabulka hran), pevné očíslování hran prvními m přirozenými čísly (pole hran)