21. - Typy datových struktur

Pole

Umožňuje přidávat prvky za sebou a přistupovat k nim pomocí indexů. Fixní velikost, kterou předem nastavíme, čímž se v paměti alokuje místo pro vstupní data.

ArrayList / List

Nemá fixní velikost, je naopak tvořen tak, že při přidání prvku se vytvoří nové pole s fixní velikostí, do kterého se staré pole zkopíruje a přidá nový prvek. Toto je nevýhodné v případě větších dat.

Java má předpřipravený ArrayList a dá se importovat.

Linked List

Linkedlist staví na tom, že máme třídu Node. Tato třída má hodnotu value a next. Next je vždy další Node, tudíž nadcházející prvek. Potom máme ještě třídu List, která má head a tail. Head je nejnovější a tail je poslední. Nevýhodou je, že pokud chceme prostřední prvek, musíme jít postupně od headu až na prostřední prvek, což může být zdlouhavé.

Časová složitost proto bude lineární.

Java má LinkedList už vytvoření, stejně jako ArrayList a dá se importnout

Kruhový buffer? :o

Implementace funguje stejně jako Linked List. Je ale potřeba, pokud se naplní, ho zase zkopírovat. Stejně jako linked list je ovšem potřeba vytvořit třídu RingBuffer, head, tail, ale pracuje jako pole. Je to tedy alternativa mezi polem a Linked Listem.

Strom

Stromy jsou podobné linked listům. Jedná se o Node, které spolu mají spojení. Node může mít více dětí. Každé dítě má nějakou hodnotu. Důležité je, že ve stromu se nemůže stát, že by nějaké dítě odkazovalo přímo na rodiče, pouze rodič odkazuje na dítě. V opačném případě by se totiž jednalo o graf. První prvek, případně nultý prvek, je zván root.

Binární strom znamená, že má rodič dvě a nebo žádné dítě.

Stromy mají hloubku a čím více je nakloněný na jednu stranu, tím méně jsou vhodné algoritmy na procházení.

Strom je skvělý pro tvoření stavových prostorů a nebo je možné využít ho k rozhodovacím stromům ve strojovém učení

Grafy

Grafy jsou podobné jako stromy, akorát je možné mít navíc jako dítě vztah zpět na rodiče. Jednotlivé body jsou zvány vrcholy. Mezi body v případě spojitosti je hrana. Grafy jsou výhodné k prohledávání například nejkratší cesty. Velké množství algoritmů nejde naprogramovat jinak než exponenciálně bruteforcem.

V programovacím jazyce je možné vytvořit toto pomocí matice a dvojrozměrného pole – jednoho pole, které má v sobě pro každý vrchol. Toto je výhodné v případě, že máme hodně hran mezi vrcholy, aby nemusela matice zbytečně obsahovat mnoho nul

List, druhá verze, naopak v sobě obsahuje seznam pouze těch vrcholů, které mají nějakou hranu. List v sobě má třídu Vrchol, která v sobě obsahuje list hran, které mají nějakou délku a šířku.

Stack / Zásobník

Zásobník se využívá pro dočasné ukládání dat. Jeho charakteristickým rysem je manipulace s daty tak, že poslední vložený prvek bude zároveň vyndán jako poslední (LIFO). Při manipulaci se udržuje ukazatel, který ukazuje poslední prvek, tudíž vršek zásobníku.

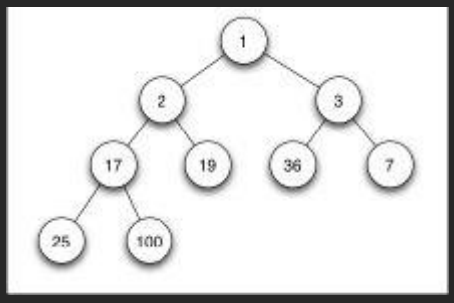
Využívá se například v Javě, kde je součástí paměti a jsou na něm uloženy reference na objekty.

Fronta

Fronta na druhé straně od zásobníku funguje na principu FIFO – First in, First Out. Prvky jsou tedy vyndávany postupně tak, jak tam byly vloženy. Fronta jako taková se využívá například v čekacích frontách, ve které jsou uloženy postupně instrukce nebo požadavky v nějakém postupu, řízení přístupu k určitému zařízené (Tiskárna), kde druhý bude muset počkat, než se obslouží první, nebo například ještě synchronizace vláken.

Halda

Halda je binární strom, kde každý uzel má přesně dva nebo žádného potomka. (Definice binárního stromu)



Jsou dva typy – Max halda nebo Min halda, kde min jde od nejmenšího prvku jako kořene a max halda bere jako hlavní kořen největší prvek.

Prvky na Haldě jsou unikátní, jinak by nebylo možné je řadit.

Haldy se využívají například u algoritmů pro řazení, jako je Heapsort nebo Priority Queue Sort, nebo vyhledávání nejkratší cesty.