

Datové struktury

pole, seznam, záznam, hash, strom, zásobník, fronta

Adamus Colepaticius Trolo

24. 1. 2025

GEVO

Obsah

Datová struktura obecně	1
Co to je datová struktura	2
Operace na datové struktuře	3
Konkrétní struktury	4
Pole (Array)	5
Seznam (List)	7
Záznam (Record, Struct)	9
Hash Table	11



Co to je datová struktura

Datová struktura je zkrátka řád uložení velkého množství souvisejících dat ve vnitřní paměti.

Při jejím návrhu pracujeme téměř výhradně s von Neumannovým modelem počítače a uvažujeme, že vnitřní paměť je **random access**, tedy na danou adresu je možný okamžitý přístup.

Podle řešeného problému volíme datovou strukturu tak, aby **nejčastější operace** trvaly, co nejkratčeji.

Operace na datové struktuře

Budeme datové struktury hodnotit z hlediska rychlosti provedení následujících operací:

- uložení (tedy i přepsání) hodnoty na danou pozici
- přečtení hodnoty na dané pozici
- přidání hodnoty na jeden z konců
- odebrání hodnoty z jednoho z konců
- přidání hodnoty na libovolnou pozici
- odebrání hodnoty z libovolné pozice
- nalezení prvku s danou hodnotou

Obsah

Datová struktura obecně	1
Co to je datová struktura	2
Operace na datové struktuře	3
Konkrétní struktury	4
Pole (Array)	5
Seznam (List)	7
Záznam (Record, Struct)	9
Hash Table	11

Pole (Array)

Pole je jednoduchá datová struktura **s danou délkou**, jež ukládá hodnoty do vnitřní paměti bezprostředně za sebou. Přístup k hodnotám probíhá přes **indexy** – vlastně počet míst v paměti od začátku pole.

		ADRESY V PAMĚTI									
		776	777	779	780	781	782	783	784	785	786
HODNOTY		8	25	43	62	42	63	41	18	42	17
	INDEXY V POLI VELIKOSTI 10	0	1	2	3	4	5	6	7	8	9

Pole (Array)

Náročnost operací

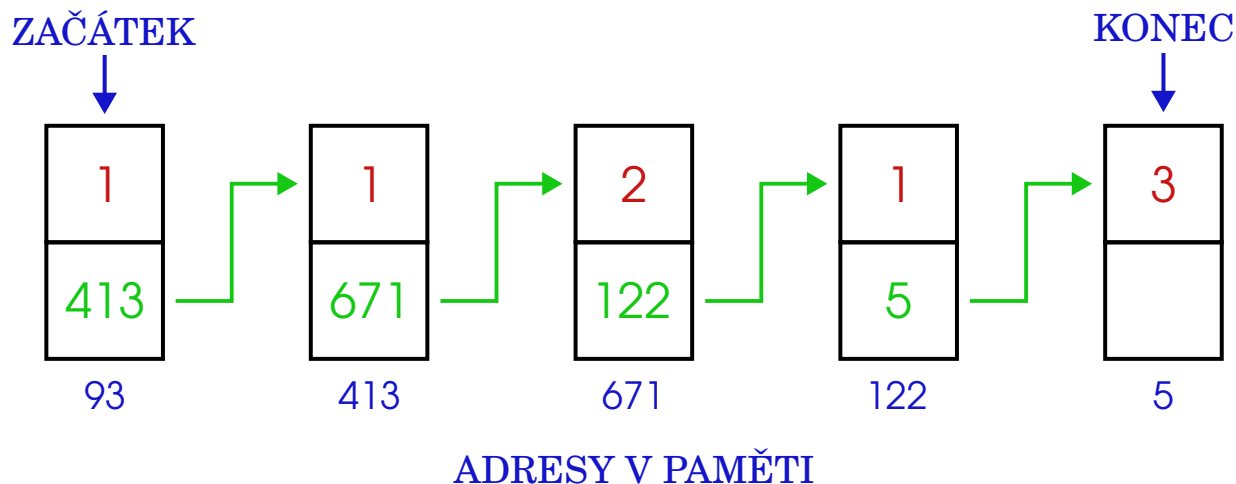
- uložení, přepis i přečtení hodnoty podle indexu: **instantní**
 - Zkrátka zapíšu hodnotu do RAM na **začátek pole + index**.
- odebrání hodnoty ze zadního konce: **instantní**
 - Zmenším velikost pole o 1.
- odebrání hodnoty na jiné pozici: **úměrné délce pole**
 - Po odebrání je potřeba přesunout všechny hodnoty s vyšším indexem o jeden index doleva.
- přidání prvku kamkoliv: **úměrné délce pole**
 - Může se stát, že pole kolem sebe nemá v paměti místo, takže je potřeba je **překopírovat jinam**.
- nalezení konkrétní hodnoty: **úměrné délce pole**
 - Prostě musím celé pole projít hodnotu po hodnotě.

Seznam (List)

Datová struktura složená z **uzlů**. Každý uzel obsahuje dvě data:

- **hodnotu**,
- **adresu v paměti** s následujícím uzlem.

Uzly nemusejí být v paměti seřazeny za sebou.



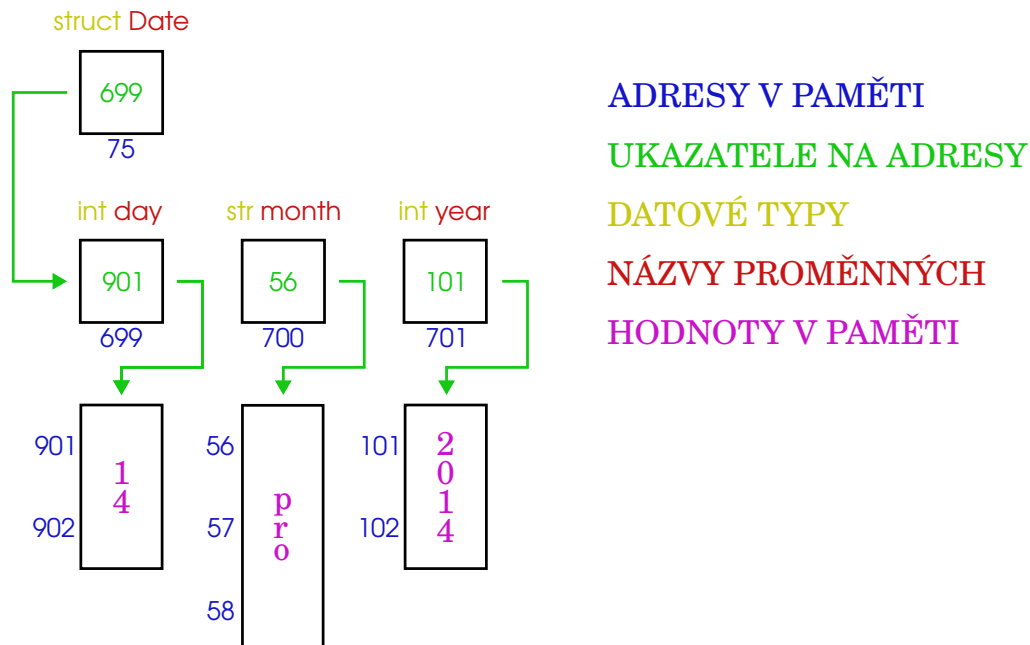
Seznam (List)

Náročnost operací

- uložení, přepis i přečtení hodnoty podle pozice: **úměrné délce seznamu**
 - Musím procházet seznam od **začátku**, dokud se nedostanu na danou pozici.
- přidání na jeden z konců / odebrání z jednoho z konců: **instantní**
 - Stačí připojit další uzel a upravit ten předchozí / následující.
- přidání / odebrání podle pozice: **úměrné délce seznamu**
 - Samotný proces přidání / odebrání je instantní, ale musím se na danou pozici nejprve dostat.
- nalezení konkrétní hodnoty: **úměrné délce seznamu**
 - Musím seznam procházet od začátku, dokud hodnotu nenajdu.

Záznam (Record, Struct)

Datová struktura obsahující množství **pojmenovaných** údajů často různých datových typů. Jména údajů jsou vlastně **proměnné uchovávající adresu v paměti**, kde začíná příslušná část záznamu.



Záznam (Record, Struct)

Náročnost operací

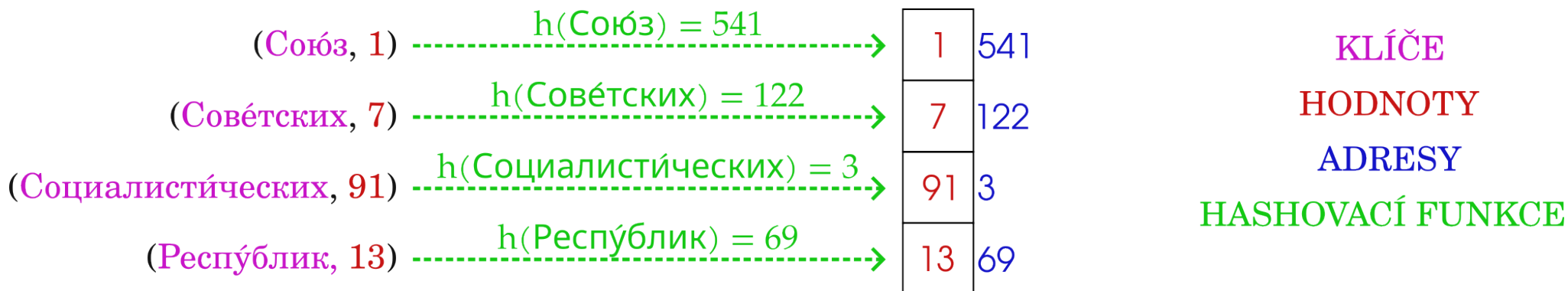
- uložení / přepis / přečtení hodnoty podle jména: **instantní**
 - Proměnná s daným jménem prostě ukazuje na adresu v RAM.
- přidání / odebrání na koncích: **nedává smysl**
 - Záznam nemá konce lol.
- přidání / odebrání podle jména: **nelze**
 - Některé struktury jako třeba dict v Pythonu přidání umožňují, ale v principu **nelze** proměnné záznamu mazat, upravovat ani přidávat.
- nalezení konkrétní hodnoty: **úměrné počtu proměnných v záznamu**
 - Záznam musím procházet proměnnou po proměnné a hledat hodnotu.

Hash Table

Struktura stvořená pro okamžité nalezení dané hodnoty podle klíče.

Pomocí předem dané „**hashovací**“ **funkce** převádí hodnoty **klíče** na **adresy** v paměti, kam potom ukládá **hodnoty**.

Hashovací funkce je **málokdy prostá**; vzniklé kolize se řeší různě, například řetězením hashovacích funkcí.



Hash Table

Náročnost operací

- uložení / přečtení / přepis hodnoty podle klíče: **instantní**
 - Spočívá v aplikaci hashovací funkce, která obvykle trvá úměrně délce klíče.
- přidání hodnoty s klíčem: **obvykle instantní**
 - Často stačí aplikace hashovací funkce. Kolize ale mohou způsobit zpomalení.
- odebrání hodnoty podle klíče: **nedává smysl**
 - Odebrání klíče nelze učinit, protože se jedná pouze o vstup do uložené hashovací funkce. Klíče samotné nikde uloženy nejsou.
- nalezení klíče: **instantní**
 - Stačí ověřit, zda je něco uloženo pod hashem klíče.
- nalezení hodnoty: **nemožné**

- ▶ Základní hash table nikde neuchovává pozice v paměti všech svých hodnot. Struktury jako třeba dict v Pythonu tenhle problém řeší ukládáním hodnot do vhodných datových struktur.