

# Algoritmy a datové struktury 1

2/2 Z+Zk, NTIN060

pomocné slajdy k datovým strukturám

Pavel Veselý (IUUK)



vesely+ads1@iuuk.mff.cuni.cz

<https://iuuk.mff.cuni.cz/~vesely/vyuka/ls24/ads1.html>

# Datové struktury

Způsob uložení dat, který umožňuje určitou sadu operací

Dva pohledy: **1.** „černá skříňka“ s daným rozhraním, tedy operacemi  
**2.** konkrétní implementace

# Datové struktury

Způsob uložení dat, který umožňuje určitou sadu operací

Dva pohledy: **1.** „černá skříňka“ s daným rozhraním, tedy operacemi  
**2.** konkrétní implementace

Příklady:

- **fronta** (*first in, first out*, FIFO) — operace Enqueue a Dequeue
- **zásobník** (*last in, first out*, LIFO) — operace Push, Pop

# Datové struktury

Způsob uložení dat, který umožňuje určitou sadu operací

Dva pohledy: **1.** „černá skříňka“ s daným rozhraním, tedy operacemi  
**2.** konkrétní implementace

Příklady:

- **fronta** (*first in, first out*, FIFO) — operace Enqueue a Dequeue
- **zásobník** (*last in, first out*, LIFO) — operace Push, Pop
- **prioritní fronta** — implementace haldou

# Datové struktury

Způsob uložení dat, který umožňuje určitou sadu operací

Dva pohledy: **1.** „černá skříňka“ s daným rozhraním, tedy operacemi  
**2.** konkrétní implementace

Příklady:

- **fronta** (*first in, first out*, FIFO) — operace Enqueue a Dequeue
- **zásobník** (*last in, first out*, LIFO) — operace Push, Pop
- **prioritní fronta** — implementace haldou
- problém **Union-Find** (udržování komponent souvislosti grafu)

# Datové struktury

Způsob uložení dat, který umožňuje určitou sadu operací

Dva pohledy: 1. „černá skříňka“ s daným rozhraním, tedy operacemi  
2. konkrétní implementace

- Příklady:
- fronta (*first in, first out*, FIFO) — operace Enqueue a Dequeue
  - zásobník (*last in, first out*, LIFO) — operace Push, Pop
  - prioritní fronta — implementace haldou
  - problém Union-Find (udržování komponent souvislosti grafu)
  - reprezentace množiny klíčů  $X \subseteq \mathcal{U}$ , kde je  $\mathcal{U}$  je univerzum
    - operace: Insert, Delete, IsMember?

# Datové struktury

Způsob uložení dat, který umožňuje určitou sadu operací

Dva pohledy: 1. „černá skříňka“ s daným rozhraním, tedy operacemi  
2. konkrétní implementace

- Příklady:
- fronta (*first in, first out*, FIFO) — operace Enqueue a Dequeue
  - zásobník (*last in, first out*, LIFO) — operace Push, Pop
  - prioritní fronta — implementace haldou
  - problém Union-Find (udržování komponent souvislosti grafu)
  - reprezentace množiny klíčů  $X \subseteq \mathcal{U}$ , kde je  $\mathcal{U}$  je univerzum
    - operace: Insert, Delete, IsMember?
  - slovník — množina dvojic  $(k, v)$ , kde  $k$  je klíč a  $v$  hodnota

# Datové struktury

Způsob uložení dat, který umožňuje určitou sadu operací

Dva pohledy: 1. „černá skříňka“ s daným rozhraním, tedy operacemi  
2. konkrétní implementace

- Příklady:
- fronta (*first in, first out*, FIFO) — operace Enqueue a Dequeue
  - zásobník (*last in, first out*, LIFO) — operace Push, Pop
  - prioritní fronta — implementace haldou
  - problém Union-Find (udržování komponent souvislosti grafu)
  - reprezentace množiny klíčů  $X \subseteq \mathcal{U}$ , kde je  $\mathcal{U}$  je univerzum
    - operace: Insert, Delete, IsMember?
  - slovník — množina dvojic  $(k, v)$ , kde  $k$  je klíč a  $v$  hodnota
  - uspořádaná množina / slovník — navíc: Min, Max, Předchůdce, atd.



# Datové struktury

Způsob uložení dat, který umožňuje určitou sadu operací

Dva pohledy: 1. „černá skříňka“ s daným rozhraním, tedy operacemi  
2. konkrétní implementace

- Příklady:
- fronta (*first in, first out*, FIFO) — operace Enqueue a Dequeue
  - zásobník (*last in, first out*, LIFO) — operace Push, Pop
  - prioritní fronta — implementace haldou
  - problém Union-Find (udržování komponent souvislosti grafu)
  - reprezentace množiny klíčů  $X \subseteq \mathcal{U}$ , kde je  $\mathcal{U}$  je univerzum
    - operace: Insert, Delete, IsMember?
  - slovník — množina dvojic  $(k, v)$ , kde  $k$  je klíč a  $v$  hodnota
  - uspořádaná množina / slovník — navíc: Min, Max, Předchůdce, atd.

- Příklady implementací:
- pole, spojový seznam, binární halda
  - binární vyhledávací strom (BVS)
  - hešovací tabulka

# Binární vyhledávací stromy (BVS)

Binary search tree (BST)

- pro problém uspořádaného slovníku (nebo uspořádané množiny klíčů)

# Binární vyhledávací stromy (BVS)

## Binary search tree (BST)

- pro problém uspořádaného slovníku (nebo uspořádané množiny klíčů)

### 1. AVL stromy = hloubkově vyvážené stromy

- G. M. **Adel'son-Velskij** a Je. M. **Landis** '62
- pro každý vrchol se hloubka podstromů levého a pravého syna liší max. o 1
- mají logaritmickou hloubku

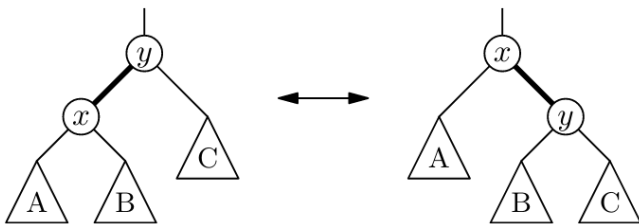
# Binární vyhledávací stromy (BVS)

## Binary search tree (BST)

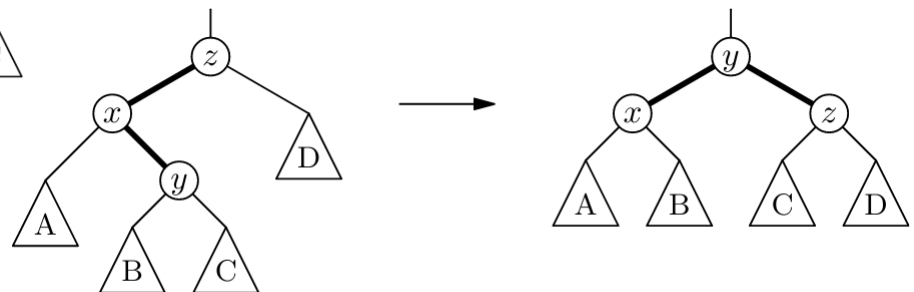
- pro problém uspořádaného slovníku (nebo uspořádané množiny klíčů)

### 1. AVL stromy = hloubkově vyvážené stromy

- G. M. **Adel'son-Velskij** a Je. M. **Landis** '62
- pro každý vrchol se hloubka podstromů levého a pravého syna liší max. o 1
- mají logaritmickou hloubku



Obrázek 8.5: Jednoduchá rotace

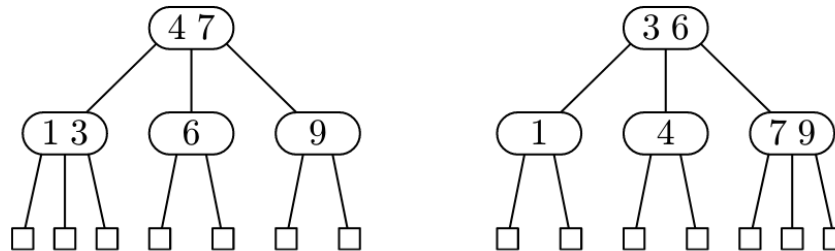


Obrázek 8.6: Dvojitá rotace

## Vícecestné vyhledávací stromy

$(a, b)$ -stromy pro  $a \geq 2, b \geq 2a - 1$

- Každý vnitřní vrchol má  $a$  až  $b$  synů, kořen má 2 až  $b$  synů
- Listy na stejné hladině a virtuální (NULL)

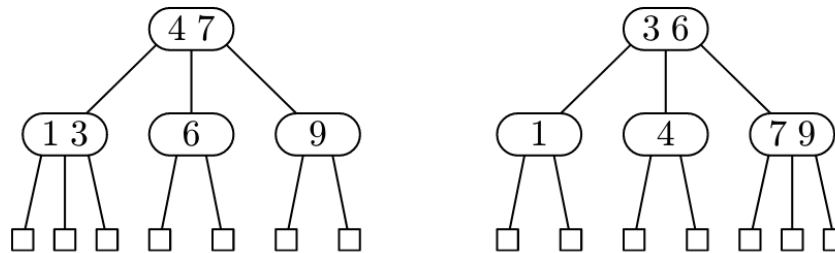


Obrázek 8.7: Dva (2,3)-stromy pro tutéž množinu klíčů

## Vícecestné vyhledávací stromy

**$(a, b)$ -stromy** pro  $a \geq 2, b \geq 2a - 1$

- Každý vnitřní vrchol má  $a$  až  $b$  synů, kořen má 2 až  $b$  synů
- Listy na stejné hladině a virtuální (NULL)



Obrázek 8.7: Dva  $(2,3)$ -stromy pro tutéž množinu klíčů

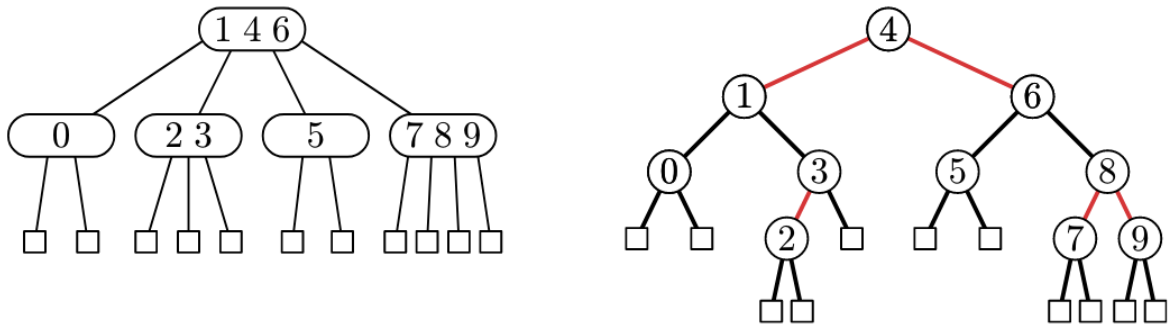
**B-stromy** [Bayer, McCreight '70]

- $(a, 2a - 1)$ - nebo  $(a, 2a)$ -stromy
- data někdy jen nejnižší hladině nad listy, vyšší hladiny obsahují pomocné klíče

# Červeno-černé stromy [Bayer '72]

varianta **left-leaning (LLRB)** [Sedgewick '08]

- překlad (2, 4)-stromů na binární stromy

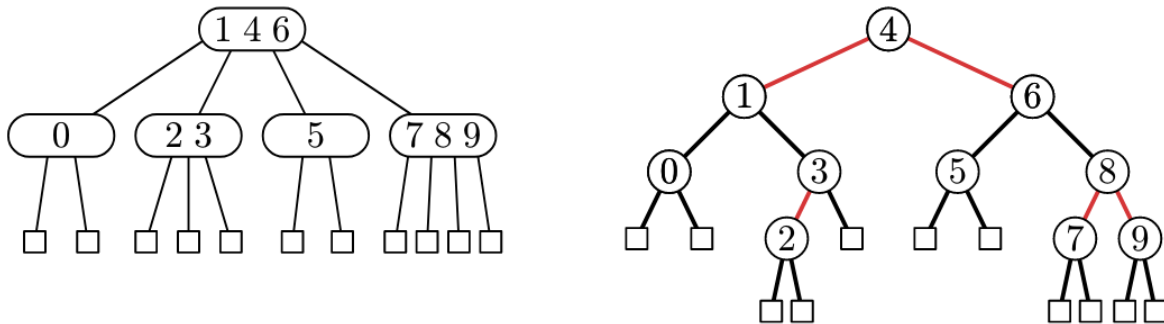


Obrázek 8.11: Překlad (2, 4)-stromu na LLRB strom

# Červeno-černé stromy [Bayer '72]

varianta **left-leaning (LLRB)** [Sedgewick '08]

- překlad (2, 4)-stromů na binární stromy



Obrázek 8.11: Překlad (2, 4)-stromu na LLRB strom

Jiná varianta **červeno-černých** stromů:

- Každý vrchol je **černý** nebo **červený**
- Kořen a (virtuální) listy jsou **černé**
- Pokud je vrchol **červený**, oba jeho synové jsou **černí**
- Pro lib. vrchol  $v$  platí, že každá cesta z  $v$  do listu má stejný # **černých** vrcholů.



## Binární vyhledávací stromy: shrnutí

- AVL stromy: menší hloubka než červeno-černé stromy, náročnější udržovat
- $(a, b)$ -stromy: vhodné pro více úrovní paměti, popř. databáze
- Červeno-černé stromy: lze dostat  $O(1)$  rotací při mazání

## Binární vyhledávací stromy: shrnutí

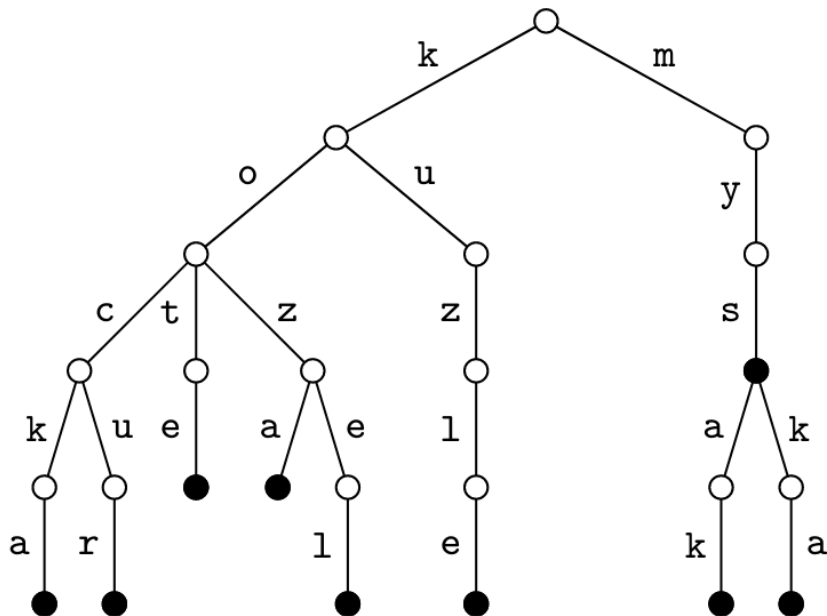
- AVL stromy: menší hloubka než červeno-černé stromy, náročnější udržovat
- $(a, b)$ -stromy: vhodné pro více úrovní paměti, popř. databáze
- Červeno-černé stromy: lze dostat  $O(1)$  rotací při mazání

## Splay stromy

- **nevyvážený** BVS
- při přístupu na vrchol vyrotujeme vrchol do kořene
- preferujeme dvojité rotace
- amortizovaná složitost  $O(\log n)$

## Písmenkový strom (trie, prefixový strom)

- trie: ze slov *tree* a *retrieval*



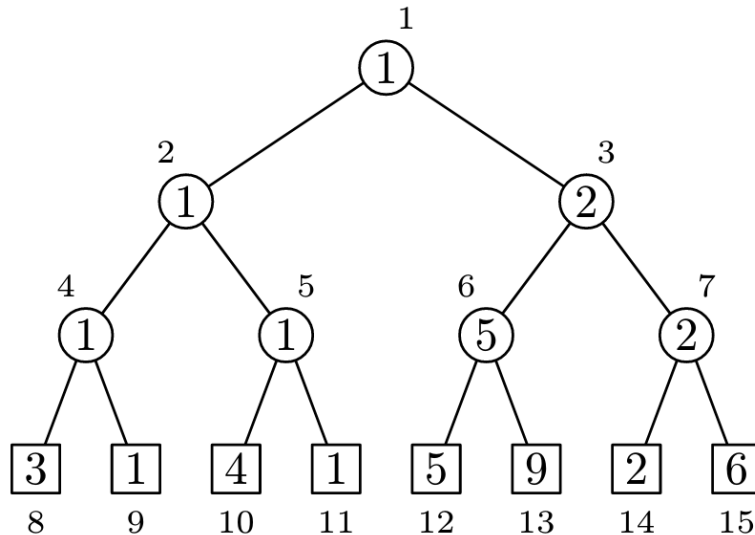
Obrázek 4.4: Písmenkový strom pro slova kocka, kocur, kote, koza, kozel, kuzle, mys, mysak, myska

## Intervalový strom

- dána posloupnost čísel  $x_0, \dots, x_{n-1}$
- zodpovídá intervalové dotazy na součet / minimum / maximum / ...
- umí zpracovat aktualizaci hodnot posloupnosti
  - včetně změny všech hodnot v intervalu (líně)

## Intervalový strom

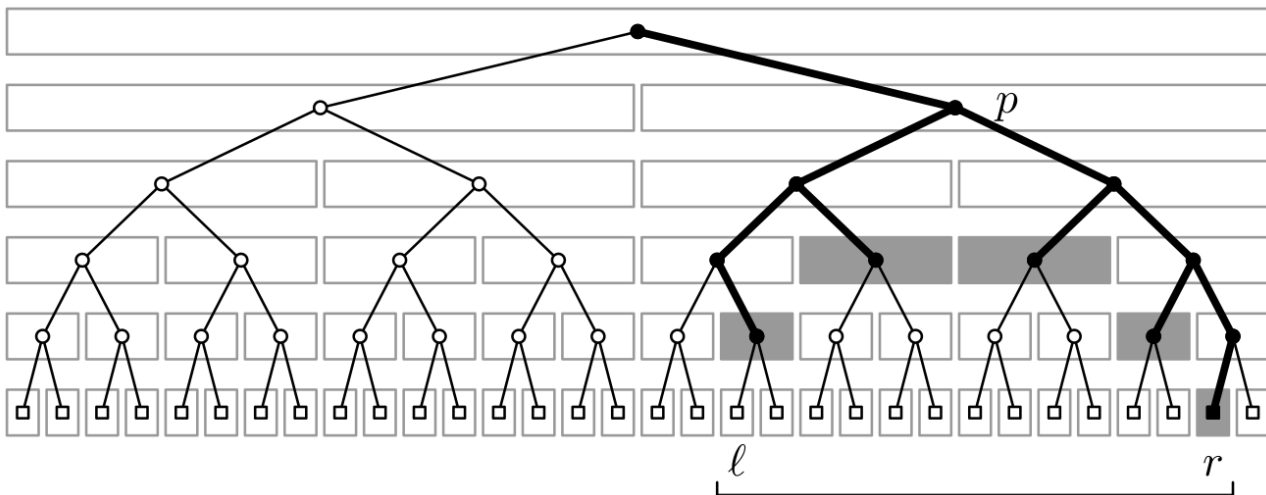
- dána posloupnost čísel  $x_0, \dots, x_{n-1}$
- zodpovídá intervalové dotazy na součet / minimum / maximum / ...
- umí zpracovat aktualizaci hodnot posloupnosti
  - včetně změny všech hodnot v intervalu (líně)



Obrázek 4.6: Intervalový strom a jeho očíslování

## Intervalový strom

- dána posloupnost čísel  $x_0, \dots, x_{n-1}$
- zodpovídá intervalové dotazy na součet / minimum / maximum / ...
- umí zpracovat aktualizaci hodnot posloupnosti
  - včetně změny všech hodnot v intervalu (líně)



Obrázek 4.7: Rozklad dotazu na kanonické intervaly

## Hešování (Hašování, Hashování) — Hashing

**Hešovací tabulka:** DS pro reprezentaci množiny (slovníku) **bez uspořádání**

- pokud nepotřebujeme hledat  $k$ -tý nejmenší klíč, součet hodnot klíčů v intervalu, apod.

## Hešování (Hašování, Hashování) — Hashing

**Hešovací tabulka:** DS pro reprezentaci množiny (slovníku) **bez uspořádání**

- pokud nepotřebujeme hledat  $k$ -tý nejmenší klíč, součet hodnot klíčů v intervalu, apod.
- „náhodnou“ funkcí  $h$  si zobrazíme množinu  $X \subseteq \mathcal{U}$  do **hešovací tabulky**
  - hešovací tabulka  $\sim$  pole s  $m$  přihrádkami, kde  $m = \Theta(n)$



## Hešování (Hašování, Hashování) — Hashing

**Hešovací tabulka:** DS pro reprezentaci množiny (slovníku) **bez uspořádání**

- pokud nepotřebujeme hledat  $k$ -tý nejmenší klíč, součet hodnot klíčů v intervalu, apod.
- „náhodnou“ funkcí  $h$  si zobrazíme množinu  $X \subseteq \mathcal{U}$  do **hešovací tabulky**
  - hešovací tabulka  $\sim$  pole s  $m$  přihrádkami, kde  $m = \Theta(n)$
- **kolize** = dva prvky z  $X$  skončí ve stejné buňce

## Hešování (Hašování, Hashování) — Hashing

**Hešovací tabulka:** DS pro reprezentaci množiny (slovníku) **bez uspořádání**

- pokud nepotřebujeme hledat  $k$ -tý nejmenší klíč, součet hodnot klíčů v intervalu, apod.
- „náhodnou“ funkcí  $h$  si zobrazíme množinu  $X \subseteq \mathcal{U}$  do **hešovací tabulky**
  - hešovací tabulka  $\sim$  pole s  $m$  přihrádkami, kde  $m = \Theta(n)$
- **kolize** = dva prvky z  $X$  skončí ve stejné buňce

Hlavní části hešovací tabulky:

### 1. Hešovací funkce

## Hešování (Hašování, Hashování) — Hashing

**Hešovací tabulka:** DS pro reprezentaci množiny (slovníku) **bez uspořádání**

- pokud nepotřebujeme hledat  $k$ -tý nejmenší klíč, součet hodnot klíčů v intervalu, apod.
- „náhodnou“ funkcí  $h$  si zobrazíme množinu  $X \subseteq \mathcal{U}$  do **hešovací tabulky**
  - hešovací tabulka  $\sim$  pole s  $m$  přihrádkami, kde  $m = \Theta(n)$
- **kolize** = dva prvky z  $X$  skončí ve stejné buňce

Hlavní části hešovací tabulky:

1. **Hešovací funkce**

2. Řešení **kolizí**:

- **Separované řetízky** — spojový seznam v každé buňce
  - průměrná délka řetízku je  $n/m$ , stejně tak střední hodnota
  - hledání, přidávání a mazání *průměrně* v čase  $O(1 + n/m)$

## Hešování (Hašování, Hashování) — Hashing

**Hešovací tabulka:** DS pro reprezentaci množiny (slovníku) **bez uspořádání**

- pokud nepotřebujeme hledat  $k$ -tý nejmenší klíč, součet hodnot klíčů v intervalu, apod.
- „náhodnou“ funkcí  $h$  si zobrazíme množinu  $X \subseteq \mathcal{U}$  do **hešovací tabulky**
  - hešovací tabulka  $\sim$  pole s  $m$  přihrádkami, kde  $m = \Theta(n)$
- **kolize** = dva prvky z  $X$  skončí ve stejné buňce

Hlavní části hešovací tabulky:

### 1. Hešovací funkce

### 2. Řešení kolizí:

- **Separované řetízky** — spojový seznam v každé buňce
  - průměrná délka řetízku je  $n/m$ , stejně tak střední hodnota
  - hledání, přidávání a mazání *průměrně* v čase  $O(1 + n/m)$
- **Otevřená adresace** — v buňce tabulky max. jeden klíč

## Hešování (Hašování, Hashování) — Hashing

**Hešovací tabulka:** DS pro reprezentaci množiny (slovníku) **bez uspořádání**

- pokud nepotřebujeme hledat  $k$ -tý nejmenší klíč, součet hodnot klíčů v intervalu, apod.
- „náhodnou“ funkcí  $h$  si zobrazíme množinu  $X \subseteq \mathcal{U}$  do **hešovací tabulky**
  - hešovací tabulka  $\sim$  pole s  $m$  přihrádkami, kde  $m = \Theta(n)$
- **kolize** = dva prvky z  $X$  skončí ve stejné buňce

Hlavní části hešovací tabulky:

### 1. Hešovací funkce

### 2. Řešení kolizí:

- **Separované řetízky** — spojový seznam v každé buňce
  - průměrná délka řetízku je  $n/m$ , stejně tak střední hodnota
  - hledání, přidávání a mazání *průměrně* v čase  $O(1 + n/m)$

- **Otevřená adresace** — v buňce tabulky max. jeden klíč

### 3. Přehešování: pokud $n/m$ příliš vzroste, zvětšíme $m$ dvakrát

- amortizovaná složitost  $O(1)$  na vložení prvku
- lze také přehešovávat, pokud je  $n/m$  příliš malé

## Hešování a narozeniny

Nechť  $h$  je plně náhodná (každý prvek z  $\mathcal{U}$  zahešován uniformně a nezávisle na ostatních).

Do hešovací tabulky vložíme  $n$  prvků.

Jak velké  $m$  zvolit, aby **nenastala kolize** s pravděpodobností třeba 99 %?

## Hešování a narozeniny

Nechť  $h$  je plně náhodná (každý prvek z  $\mathcal{U}$  zahešován uniformně a nezávisle na ostatních).

Do hešovací tabulky vložíme  $n$  prvků.

Jak velké  $m$  zvolit, aby **nenastala kolize** s pravděpodobností třeba 99 %?

Narozeninový paradox:

- pokud je v místnosti  $\geq 23$  lidí, s pravděpodobností  $\geq 50$  % jsou tu dva se stejnými narozeninami

## Hešování a narozeniny

Nechť  $h$  je plně náhodná (každý prvek z  $\mathcal{U}$  zahešován uniformně a nezávisle na ostatních).

Do hešovací tabulky vložíme  $n$  prvků.

Jak velké  $m$  zvolit, aby **nenastala kolize** s pravděpodobností třeba 99 %?

**Narozeninový paradox:**

- pokud je v místnosti  $\geq 23$  lidí, s pravděpodobností  $\geq 50$  % jsou tu dva se stejnými narozeninami

$\Rightarrow$  potřebovali bychom  $m \in \Omega(n^2)$  pro konstantní pravděpodobnost žádné kolize



## Otevřená adresace

- v přihrádce (buňce tabulky) max. jeden klíč
- každý klíč má **vyhledávací posloupnost** = permutaci  $[m] = \{0, 1, \dots, m - 1\}$ 
  - dána funkcí  $g(x, i)$ , kde  $x$  je klíč a  $i \in [m]$

## Otevřená adresace

- v přihrádce (buňce tabulky) max. jeden klíč
- každý klíč má **vyhledávací posloupnost** = permutaci  $[m] = \{0, 1, \dots, m - 1\}$ 
  - dána funkcí  $g(x, i)$ , kde  $x$  je klíč a  $i \in [m]$

### 1. Lineární přidávání: $g(x, i) = (h(x) + i) \bmod m$

- $h : \mathcal{U} \rightarrow [m]$  je „běžná“ hešovací funkce
- **Výhoda:** sekvenční přístup do paměti (tj. dobře se chová ke keším CPU)
- **Nevýhoda:** tvoří se souvislé bloky obsazených buněk

### 2. Dvojitě hešování: $g(x, i) = (h_1(x) + h_2(x)) \cdot i \bmod m$

- $h_1$  a  $h_2$  jsou hešovací funkce z  $\mathcal{U}$  do  $[m]$
- $h_2(x)$  a  $m$  musí být **nesoudělné**  $\Rightarrow$  hodí se mít  $m$  prvočíselné
- opačné (ne)výhody oproti lineárnímu přidávání

## Otevřená adresace

- v přihrádce (buňce tabulky) max. jeden klíč
- každý klíč má **vyhledávací posloupnost** = permutaci  $[m] = \{0, 1, \dots, m - 1\}$ 
  - dána funkcí  $g(x, i)$ , kde  $x$  je klíč a  $i \in [m]$

### 1. Lineární přidávání: $g(x, i) = (h(x) + i) \bmod m$

- $h : \mathcal{U} \rightarrow [m]$  je „běžná“ hešovací funkce
- **Výhoda:** sekvenční přístup do paměti (tj. dobře se chová ke keším CPU)
- **Nevýhoda:** tvoří se souvislé bloky obsazených buněk

### 2. Dvojitě hešování: $g(x, i) = (h_1(x) + h_2(x)) \cdot i \bmod m$

- $h_1$  a  $h_2$  jsou hešovací funkce z  $\mathcal{U}$  do  $[m]$
- $h_2(x)$  a  $m$  musí být **nesoudělné**  $\Rightarrow$  hodí se mít  $m$  prvočíselné
- opačné (ne)výhody oproti lineárnímu přidávání

## Jak mazat?

- Nelze jen smazat klíč — hrozí přerušení vyhledávací posloupnosti jiného klíče!
- Řešení: **pomníčky** — klíč jen označíme za smazaný, buňku necháme obsazenou
  - Když je v tabulce příliš mnoho pomníčků, vše přehešujeme

## Hešovací funkce

- **Plně náhodná** (každý prvek z  $\mathcal{U}$  zahešován uniformně a nezávisle na ostatních).
  - Potřebuje prostor  $\Omega(|\mathcal{U}|)$  na uložení :-)

# Hešovací funkce

- **Plně náhodná** (každý prvek z  $\mathcal{U}$  zahešován uniformně a nezávisle na ostatních).
  - Potřebuje prostor  $\Omega(|\mathcal{U}|)$  na uložení :-)

## Praktické funkce $h$ :

- **Lineární kongruence:**  $h_{\text{lin}}(x) = a \cdot x \pmod{m}$ 
  - $a$  nesoudělné s  $m$ , např.  $m$  prvočíslo a  $a \approx 0.618 \cdot m$

# Hešovací funkce

- **Plně náhodná** (každý prvek z  $\mathcal{U}$  zahešován uniformně a nezávisle na ostatních).
  - Potřebuje prostor  $\Omega(|\mathcal{U}|)$  na uložení :-)

## Praktické funkce $h$ :

- **Lineární kongruence:**  $h_{\text{lin}}(x) = a \cdot x \pmod{m}$ 
  - $a$  nesoudělné s  $m$ , např.  $m$  prvočíslo a  $a \approx 0.618 \cdot m$
- **Vyšší bity součinu** (multiply-shift):  $h_{\text{MS}}(x) = (a \cdot x \pmod{2^w}) \gg (w - \ell)$ 
  - $a$  liché,  $m = 2^\ell$ . Předpokládáme, že  $x, a$  jsou  $w$ -bitová čísla

# Hešovací funkce

- **Plně náhodná** (každý prvek z  $\mathcal{U}$  zahešován uniformně a nezávisle na ostatních).
  - Potřebuje prostor  $\Omega(|\mathcal{U}|)$  na uložení :-)

## Praktické funkce $h$ :

- **Lineární kongruence:**  $h_{\text{lin}}(x) = a \cdot x \pmod{m}$ 
  - $a$  nesoudělné s  $m$ , např.  $m$  prvočíslo a  $a \approx 0.618 \cdot m$
- **Vyšší bity součinu** (multiply-shift):  $h_{\text{MS}}(x) = (a \cdot x \pmod{2^w}) \gg (w - \ell)$ 
  - $a$  liché,  $m = 2^\ell$ . Předpokládáme, že  $x, a$  jsou  $w$ -bitová čísla
- **Skalární součin:**  $h_{\text{skal}}((x_0, \dots, x_{d-1})) = \left( \sum_{i=0}^{d-1} a_i x_i \right) \pmod{m}$ 
  - parametry  $a_0, \dots, a_{d-1}$  (měly by být různé)

# Hešovací funkce

- **Plně náhodná** (každý prvek z  $\mathcal{U}$  zahešován uniformně a nezávisle na ostatních).
  - Potřebuje prostor  $\Omega(|\mathcal{U}|)$  na uložení :-)

## Praktické funkce $h$ :

- **Lineární kongruence:**  $h_{\text{lin}}(x) = a \cdot x \pmod{m}$ 
  - $a$  nesoudělné s  $m$ , např.  $m$  prvočíslo a  $a \approx 0.618 \cdot m$
- **Vyšší bity součinu** (multiply-shift):  $h_{\text{MS}}(x) = (a \cdot x \pmod{2^w}) \gg (w - \ell)$ 
  - $a$  liché,  $m = 2^\ell$ . Předpokládáme, že  $x, a$  jsou  $w$ -bitová čísla
- **Skalární součin:**  $h_{\text{skal}}((x_0, \dots, x_{d-1})) = \left( \sum_{i=0}^{d-1} a_i x_i \right) \pmod{m}$ 
  - parametry  $a_0, \dots, a_{d-1}$  (měly by být různé)
- **Polynomiální:**  $h_{\text{poly}}((x_0, \dots, x_{d-1})) = \left( \sum_{i=0}^{d-1} a^i x_i \right) \pmod{m}$ 
  - jen jeden parametr  $a$ , hodí se mít  $m$  prvočíselné



# Hešovací funkce

- **Plně náhodná** (každý prvek z  $\mathcal{U}$  zahešován uniformně a nezávisle na ostatních).
  - Potřebuje prostor  $\Omega(|\mathcal{U}|)$  na uložení :-)

## Praktické funkce $h$ :

- **Lineární kongruence:**  $h_{\text{lin}}(x) = a \cdot x \pmod{m}$ 
  - $a$  nesoudělné s  $m$ , např.  $m$  prvočíslo a  $a \approx 0.618 \cdot m$
- **Vyšší bity součinu** (multiply-shift):  $h_{\text{MS}}(x) = (a \cdot x \pmod{2^w}) \gg (w - \ell)$ 
  - $a$  liché,  $m = 2^\ell$ . Předpokládáme, že  $x, a$  jsou  $w$ -bitová čísla
- **Skalární součin:**  $h_{\text{skal}}((x_0, \dots, x_{d-1})) = \left( \sum_{i=0}^{d-1} a_i x_i \right) \pmod{m}$ 
  - parametry  $a_0, \dots, a_{d-1}$  (měly by být různé)
- **Polynomiální:**  $h_{\text{poly}}((x_0, \dots, x_{d-1})) = \left( \sum_{i=0}^{d-1} a^i x_i \right) \pmod{m}$ 
  - jen jeden parametr  $a$ , hodí se mít  $m$  prvočíselné
- **Tabelace**
- **MurmurHash, ...**

## Univerzální a $k$ -nezávislé hešování

**Definice.** Systém funkcí  $\mathcal{H}$  z univerza  $\mathcal{U}$  do  $[m] = \{0, \dots, m - 1\}$  je  $c$ -univerzální pro  $c \geq 1$ , pokud  $\forall x, y \in \mathcal{U}, x \neq y : \Pr_{h \in \mathcal{H}} [h(x) = h(y)] \leq \frac{c}{m}$ .

- př. 1-univerzálního systému:  $\mathcal{H}_{\text{lin}} = \{h_{a,b} \mid a, b \in [p], a \neq 0\}$ 
    - kde  $h_{a,b} = ((ax + b) \bmod p) \bmod m$  a  $p \geq |\mathcal{U}|$  je prvočíslo
-

## Univerzální a $k$ -nezávislé hešování

**Definice.** Systém funkcí  $\mathcal{H}$  z univerza  $\mathcal{U}$  do  $[m] = \{0, \dots, m - 1\}$  je  $c$ -univerzální pro  $c \geq 1$ , pokud  $\forall x, y \in \mathcal{U}, x \neq y : \Pr_{h \in \mathcal{H}} [h(x) = h(y)] \leq \frac{c}{m}$ .

- př. 1-univerzálního systému:  $\mathcal{H}_{\text{lin}} = \{h_{a,b} \mid a, b \in [p], a \neq 0\}$ 
    - kde  $h_{a,b} = ((ax + b) \bmod p) \bmod m$  a  $p \geq |\mathcal{U}|$  je prvočíslo
- 

**Definice.** Systém funkcí  $\mathcal{H}$  z univerza  $\mathcal{U}$  do  $[m]$  je  $(2, c)$ -nezávislý pro  $c \geq 1$ , pokud

$$\forall x_1, x_2 \in \mathcal{U}, x_1 \neq x_2, \forall y_1, y_2 \in [m] : \Pr_{h \in \mathcal{H}} [h(x_1) = y_1 \wedge h(x_2) = y_2] \leq \frac{c}{m^2}.$$

- př.  $(2, c)$ -nezávislého systému:  $\mathcal{H}_{\text{lin}}$
  - někdy se  $(2, c)$ -nezávislosti říká silná  $c$ -univerzalita
-

## Univerzální a $k$ -nezávislé hešování

**Definice.** Systém funkcí  $\mathcal{H}$  z univerza  $\mathcal{U}$  do  $[m] = \{0, \dots, m - 1\}$  je  **$c$ -univerzální** pro  $c \geq 1$ , pokud  $\forall x, y \in \mathcal{U}, x \neq y : \Pr_{h \in \mathcal{H}} [h(x) = h(y)] \leq \frac{c}{m}$ .

- př. 1-univerzálního systému:  $\mathcal{H}_{\text{lin}} = \{h_{a,b} \mid a, b \in [p], a \neq 0\}$ 
    - kde  $h_{a,b} = ((ax + b) \bmod p) \bmod m$  a  $p \geq |\mathcal{U}|$  je prvočíslo
- 

**Definice.** Systém funkcí  $\mathcal{H}$  z univerza  $\mathcal{U}$  do  $[m]$  je  **$(2, c)$ -nezávislý** pro  $c \geq 1$ , pokud

$$\forall x_1, x_2 \in \mathcal{U}, x_1 \neq x_2, \forall y_1, y_2 \in [m] : \Pr_{h \in \mathcal{H}} [h(x_1) = y_1 \wedge h(x_2) = y_2] \leq \frac{c}{m^2}.$$

- př.  $(2, c)$ -nezávislého systému:  $\mathcal{H}_{\text{lin}}$
  - někdy se  $(2, c)$ -nezávislosti říká silná  $c$ -univerzalita
- 

**Definice.** Systém funkcí  $\mathcal{H}$  z univerza  $\mathcal{U}$  do  $[m]$  je  **$k$ -nezávislý** pro  $k \geq 2$ , pokud

$$\forall x_1, \dots, x_k \in \mathcal{U} \text{ různé}, \forall y_1, \dots, y_k \in [m] : \Pr_{h \in \mathcal{H}} [h(x_1) = y_1 \wedge \dots \wedge h(x_k) = y_k] \leq \frac{O(1)}{m^k}$$

- př.: funkce  $\left( \left( \sum_{i=0}^{k-1} a_i x^i \right) \bmod p \right) \bmod m$  a  $p \geq |\mathcal{U}|$  je prvočíslo