IB111 Programování a algoritmizace

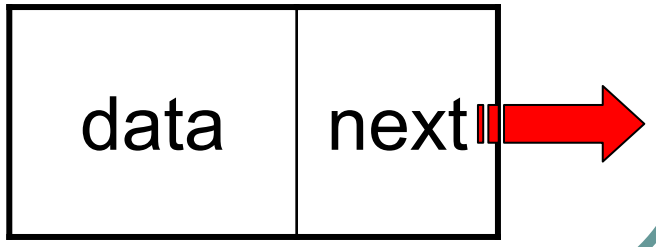
Datové struktury:

Zřetězené seznamy

Zřetězený seznam

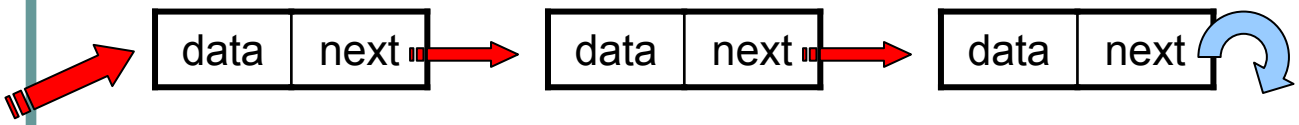
- Datová struktura spolu s ukazatelem na další prvek
  - Ukazatel ukazuje na příští nebo předchozí prvek
- class Node:

```
def __init__(self,value):
    self.data = value
    self.next = 0
```

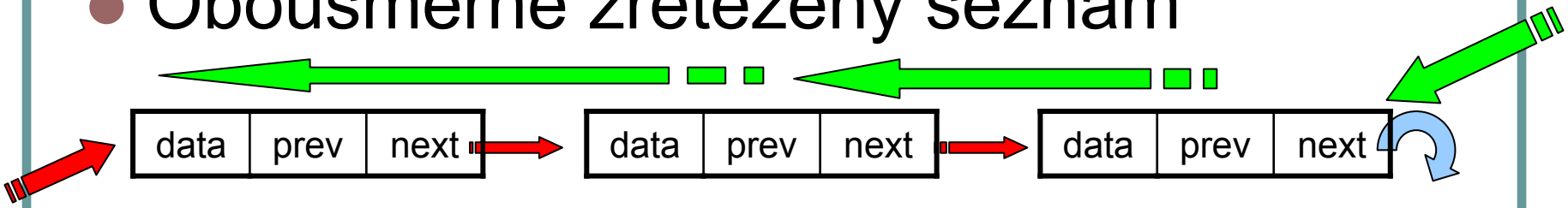


Typy zřetězených seznamů

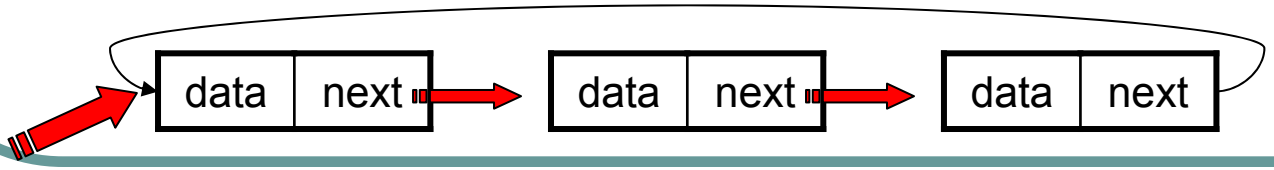
- Jednosměrně zřetězený seznam



- Obousměrně zřetězený seznam

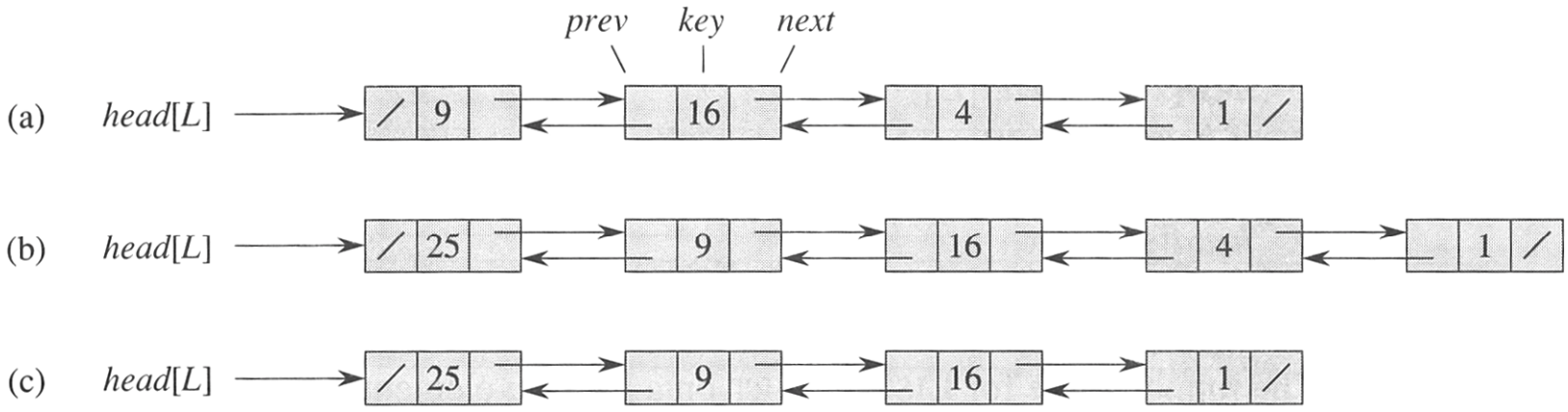


- Kruhový seznam
  - Jednosměrný nebo obousměrný



Obousměrně zřetězený seznam

- Příklad

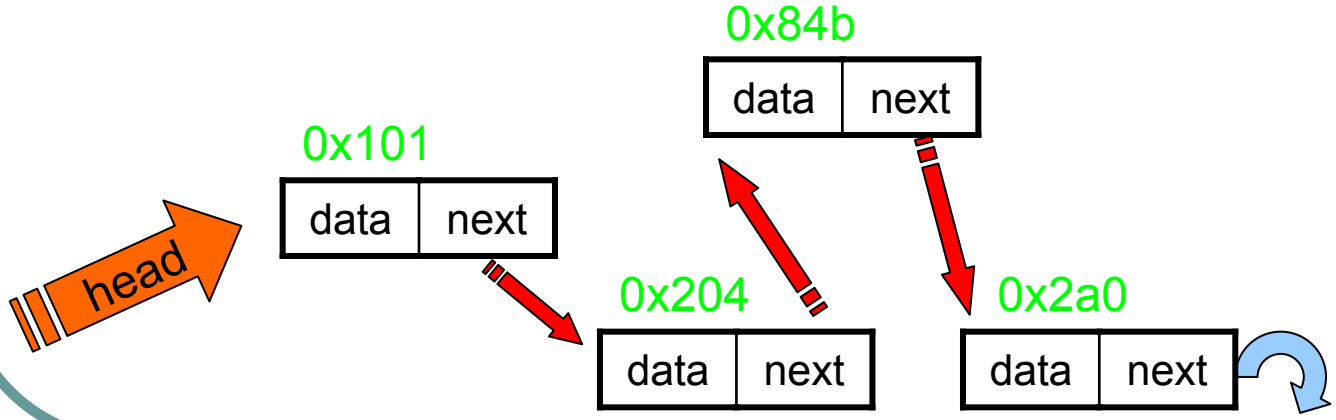


T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein: Introduction to Algorithms.

Praktický příklad v Pythonu

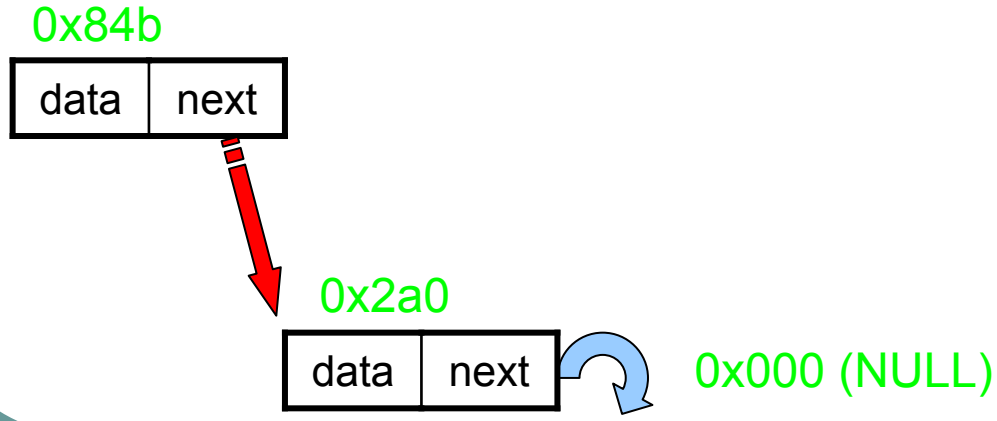
- class Node:
  def __init__(self,v,p,j):
  self.vek = v
  self.studijni_prumer = p
  self.jmeno = j
  self.next = 0
- head = Node (20, 2.1, “Jan Kos”)

Data nejsou uložena v paměti kontinuálně, ale na různých adresách podle toho jak jsme prvky přidávali/alokovali.



Konec seznamu

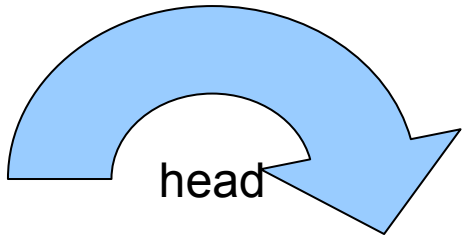
- Jak zakončit zřetězený seznam
  - Ukazatel na další prvek nastavit na 0/NULL
  - Přidat speciální prvek, který nebude obsahovat žádnou hodnotu
  - Kruhový seznam



Př. Prázdný seznam

- class Node:
  def __init__(self,v,p,j):
  self.vek = v
  self.studijni_prumer = p
  self.jmeno = j
  self.next = 0
- head = 0

```
>>> print(head)
0
```

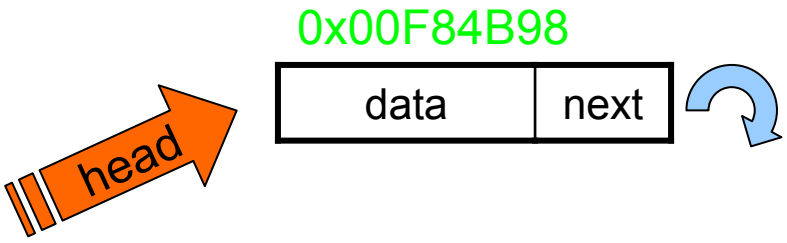


0x000 (NULL)

Př: Seznam s jedním prvkem

- head = Node (20, 2.0, “Vaclav Janik”)

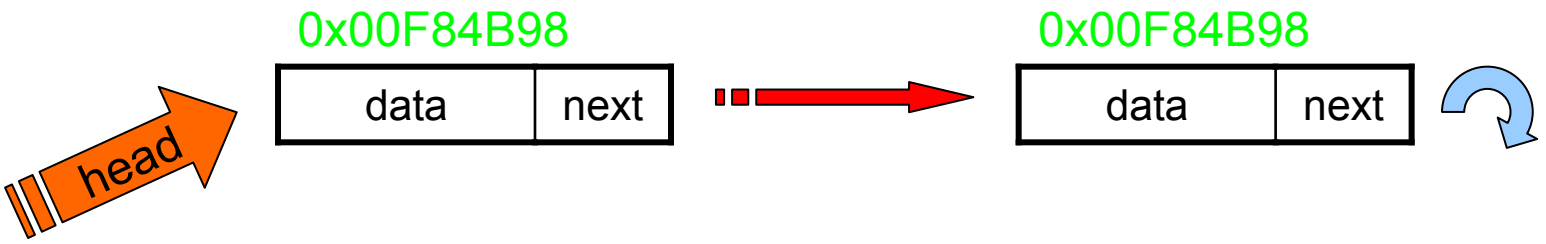
```
>>> print(head)
<__main__.Node instance at 0x00F84B98>
```



Př: přidáme druhý prvek

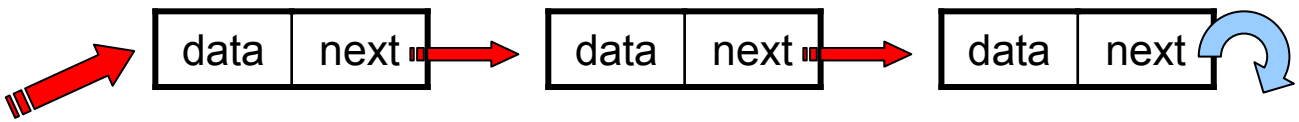
- head.next = Node(30, 1.3, "Alan")

```
>>> head.next = Node(30,1.3,"Alan")
>>> print(head.next)
<__main__.Node instance at 0x00F84B98>
>>> print(head.next.next)
0
```



Procházení seznamu

- Úkol: projít (např. vypsat) všechny prvky seznamu
  - Začneme u prvku, na který ukazuje *head*
  - Opakujeme dokud další prvek není 0/NULL
    - Zpracuj prvek (např. vypiš hodnotu data)
    - běž na další prvek



Hledání v nesetříděném seznamu

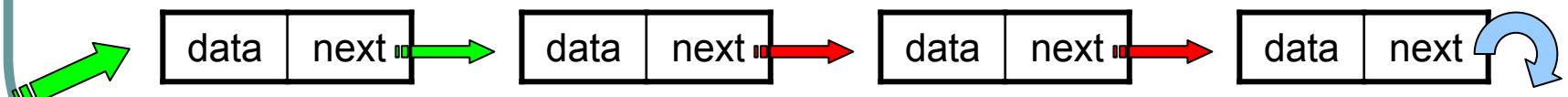
- Podobné jako procházení, ale porovnávám hodnotu s klíčem

```
LIST-SEARCH(L, k)
1  x ← head[L]
2  while x ≠ NIL and key[x] ≠ k
3       do x ← next[x]
4  return x
```

Přidávání prvku

- Alokujeme paměť pro další prvek, naplníme jeho datovou část a zařadíme do seznamu
  - Na začátek
  - Na konec
  - Časová náročnost

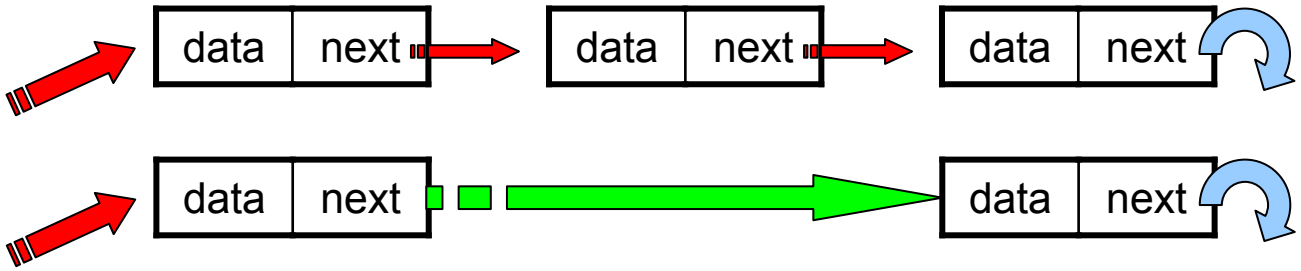
```
LIST-INSERT(L, x)
1  next[x] ← head[L]
2  if head[L] ≠ NIL
3     then prev[head[L]] ← x
4  head[L] ← x
5  prev[x] ← NIL
```



Odebrání prvku

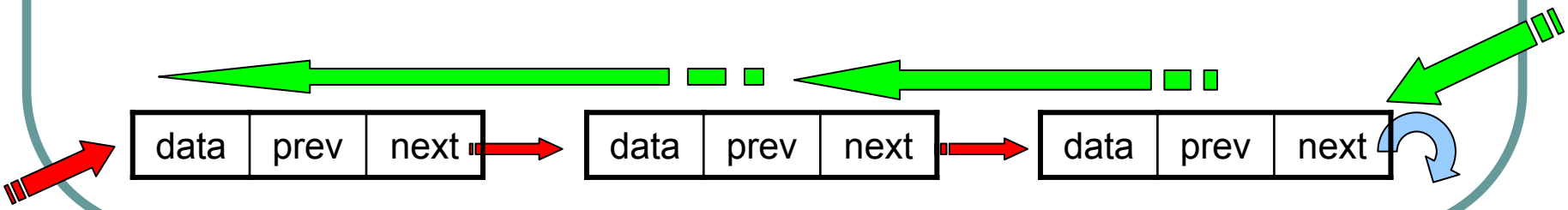
- Musíme odpojit prvek ze seznamu, dále můžeme uvolnit paměť použitou pro prvek

```
LIST-DELETE(L, x)
1  if prev[x] ≠ NIL
2       then next[prev[x]] ← next[x]
3       else head[L] ← next[x]
4  if next[x] ≠ NIL
5       then prev[next[x]] ← prev[x]
```



Oboustranně zřetězené seznamy

- Máme ukazatel na začátek a konec seznamu
  - Lehce můžeme přistoupit k prvnímu i poslednímu prvku
  - Ale musíme vždy aktualizovat oba směry
    - Přidávání prvků
    - Mazání prvků

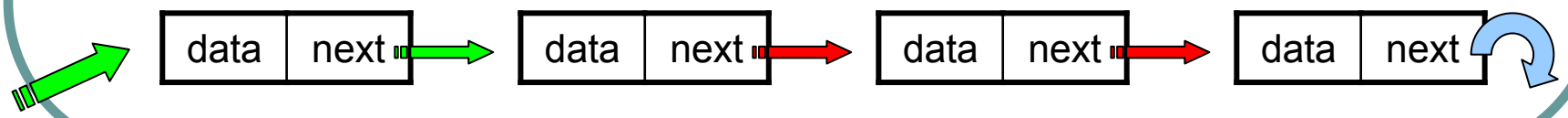


Využití spojených seznamů

- Seznam prvků
  - Přidání, mazání, vyhledávání, výpis prvků
- Zásobník (LIFO)
  - Přidání (na vrchol zásobníku), odebrání (z vrcholu zásobníku), je zásobník prázdný?
- Fronta (FIFO)
  - Přidání prvku, odebrání prvku, je fronta prázdná

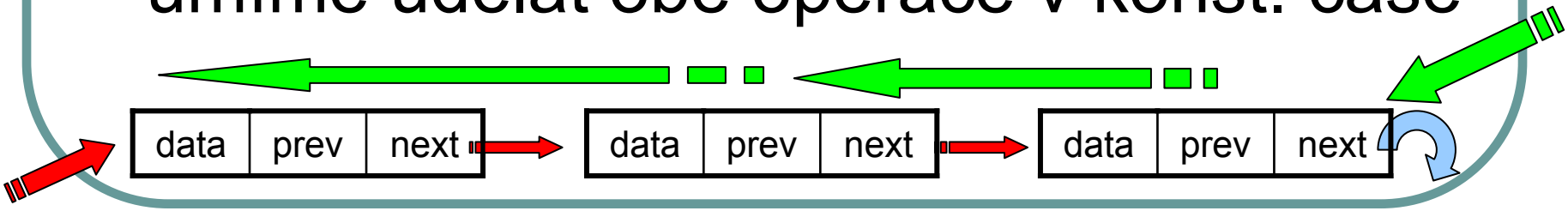
Zásobník (LIFO)

- Stačí jednosměrně zřetězený seznam
- Přidáváme na začátek seznamu
- Odebíráme ze začátku seznamu
  - Pokud je seznam neprázdný
- Obě operace v konstantním čase



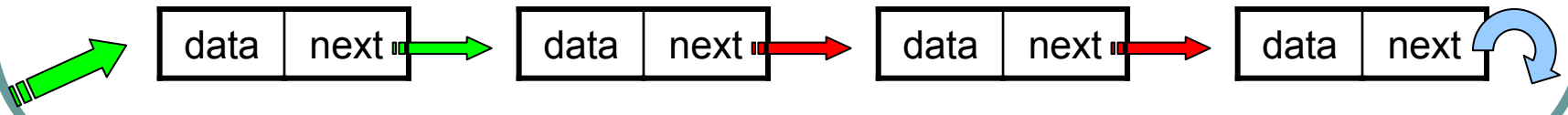
Fronta (FIFO)

- Můžeme použít jednosměrně i obousměrně zřetězený seznam
- Přidáváme na začátek a odebíráme z konce
  - Nebo obráceně
- U oboustranně zřetězeného seznamu umíme udělat obě operace v konst. čase



Fronta (FIFO)

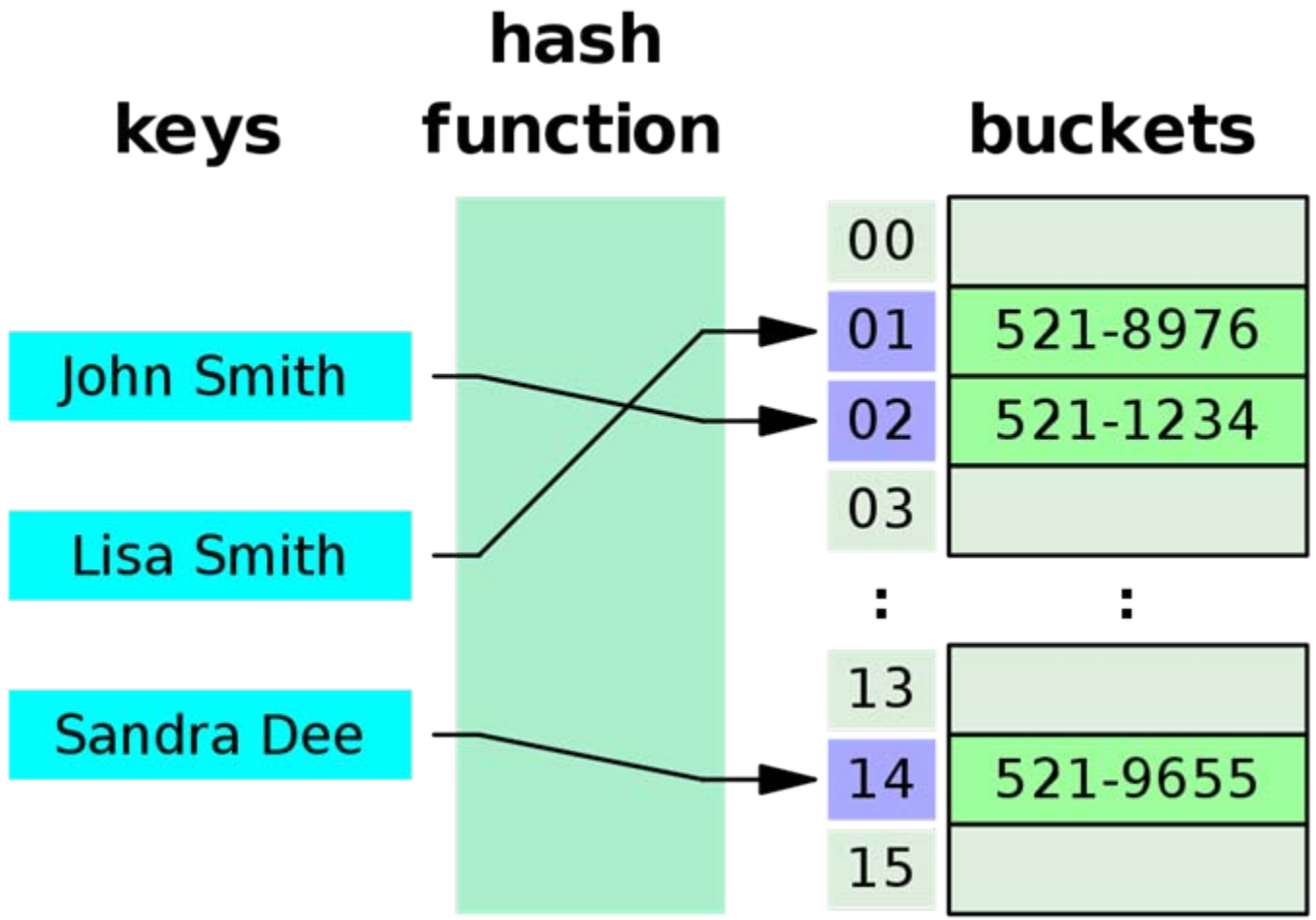
- U jednostranně zřetězeného seznamu umíme buď konstantní přidání nebo konstantní odebrání
  - Druhá operace vyžaduje lineární procházení všech záznamů
  - Můžeme také použít dodatečný pointer na poslední prvek pro přidávání na konec



Hašovací tabulka

- Datová struktura pro efektivní vyhledávání
- Asociuje klíče s odpovídajícími hodnotami
  - Hašovací funkce
- Hašovací funkce nebývá prostá
  - Vznikají kolize
    - Řešíme kolizním seznamem nebo ukládáním na další volná místa

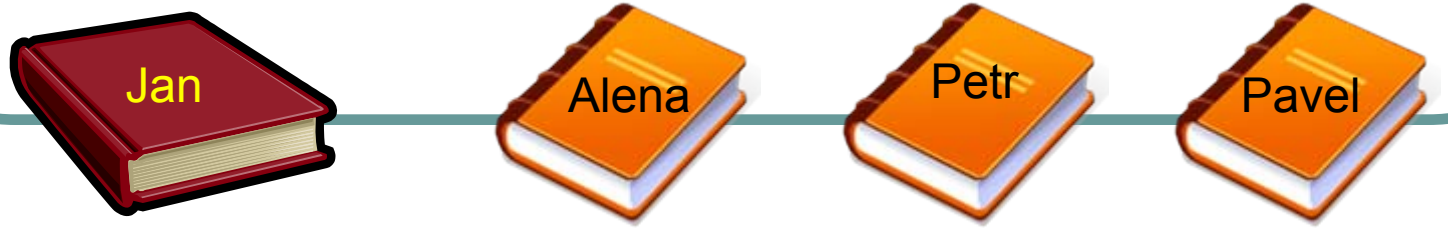
Hašovací tabulka



Zdroj: Wikipedia

Hašovací tabulka v praxi

- Knihovna
  - Objednáte knihy online, pak si pro ně přijdete
  - Knihy jsou připraveny pro řadu osob
- Knihy můžeme náhodně položit na stůl
  - Rychlé přidání
  - Pomalé hledání
    - Procházíme všechny položky



Hašovací tabulka v praxi

- Můžeme připravit poličky pro jednotlivá písmena abecedy a knihy vložíme do poličky podle prvního písmene příjmení
  - Některé poličky budou plné, některé velice málo využívané
- Můžeme připravit poličky podle posledního čísla (nebo 2 čísel) průkazy čtenáře
  - Poličky budou využívány rovnoměrněji

Příští přednáška

- Cvičení v B117 od 12:00
- Příští přednáška 8.11.2011 od 10:00