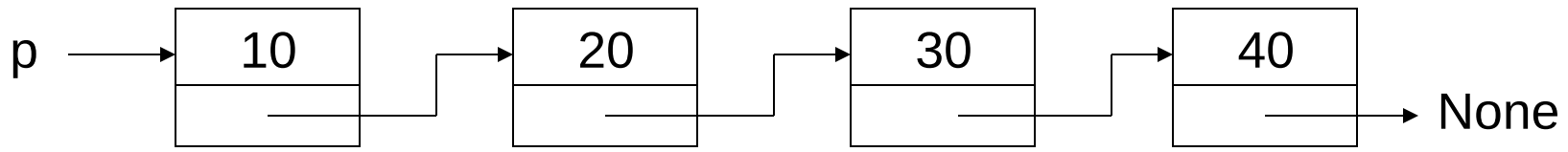
Lineární spojový seznam

```
class Uzel:
    """uzel spojového seznamu"""

    def __init__(self, x = None, dal = None):
        self.info = x              # uložená hodnota
        self.dalsi = dal           # následník
```



```
# vytvoření spojového seznamu p s hodnotami 10 20 30
p = Uzel(10)
q = Uzel(20)
r = Uzel(30)
p.dalsi = q
q.dalsi = r

p = Uzel(10, Uzel(20, Uzel(30)))

# průchod a výpis
s = p
while s != None:
    print(s.info, end = " ")
    s = s.dalsi
print()
```

```
# poslední uzel
if p == None:
    print("prazdny seznam")
else:
    s = p
    while s.dalsi != None:
        s = s.dalsi
    print(s.info)
```

```
# vyhledání zadané hodnoty
hodnota = 20
print("hledame", hodnota)
s = p
while s != None and s.info != hodnota:
    s = s.dalsi
if s == None:
    print("nenalezen")
else:
    print("nalezen", s.info)
```

```
# přidání na začátek seznamu
t = Uzel(40)
t.dalsi = p
p = t

# přidání na konec seznamu
if p == None:
    p = Uzel(50)
else:
    s = p
    while s.dalsi != None:
        s = s.dalsi
    s.dalsi = Uzel(50)
```

Operace se spojovým seznamem

- určit počet prvků
- vypsat všechny hodnoty
- nalezení posledního prvku
- vyhledání prvku s danou hodnotou

- přidání prvku na začátek, na konec seznamu
- vytvoření seznamu z dat na vstupu
- vytvoření kopie seznamu

- přidání prvku na dané místo
- přidání prvku do uspořádaného seznamu (na správné místo)

- odebrání prvku ze začátku, z konce seznamu
- odebrání daného prvku

Operace se spojovým seznamem (pokr.)

- zrušení všech prvků v seznamu s danou hodnotou
- obrácení pořadí prvků v seznamu
- uspořádání prvků v seznamu podle hodnoty

- spojení dvou seznamů do jednoho (ze sebe)
- slití dvou uspořádaných seznamů do jednoho (merge)

- rozdělení seznamu do dvou (podle pozice lichý/sudý)
- rozdělení seznamu do dvou (podle hodnoty lichý/sudý)

Příklady použití lineárních spojových seznamů

Zásobník

- realizován jednoduchým LSS, ukazatel na začátek seznamu ukazuje na vrchol zásobníku (dno zásobníku = **`None`**)
- prázdný zásobník = prázdný LSS
- přidávání i odebírání prvků se provádí **na začátku** LSS – snadné

```
class Zasobnik:
    """zásobník uložený v seznamu"""

    def __init__(self):
        self.s = []                 # prázdný zásobník

    def pridej(self, x):
        self.s.append(x)

    def odeber(self):
        return self.s.pop()
```

```
class Zasobnik:
    """zásobník jako spojový seznam"""

    def __init__(self):
        self.p = None           # prázdný zásobník

    def pridej(self, x):        # na začátek
        q = Uzel(x)
        q.dalsi = self.p
        self.p = q

    def odeber(self):           # ze začátku
        q = self.p
        self.p = self.p.dalsi
        return q.info
```

```
z = Zasobnik()
#je jedno, kterou implementaci třídy Zasobnik použijeme

z.pridej(20)
z.pridej(30)
print(z.odeber())
print(z.odeber())
```

Poznámka: v metodě odeber() jsme pro jednoduchost nekontrolovali, zda zásobník není prázdný.

Fronta

- realizována jednoduchým LSS a dvěma ukazateli:
  na začátek (tj. na první prvek LSS) a
  na konec (tj. na poslední prvek LSS)

- na začátku LSS se dobře přidává i odebírá prvek,
  na konci LSS se dobře přidává, ale špatně odebírá
  → **na začátku LSS bude odchod, na konci LSS bude příchod**

(tzn. každý ukazuje na toho, kdo přišel po něm,
což je obráceně, než ve frontách v běžném životě)

- prázdná fronta = prázdný LSS

Jiná možná realizace: LSS s hlavou
(snáze se pak programují operace s dočasně prázdnou frontou)

Dlouhá čísla

- uložena po cifrách nebo po skupinách cifer podobně jako v poli

- nejsme omezeni maximálním možným počtem cifer v čísle

- směr řazení LSS závisí na prováděných operacích,
např. pro sčítání nebo násobení odzadu od cifer nejnižšího řádu,
pro výpis ale potřebujeme procházet cifry odpředu
→ bude nutno otáčet seznam nebo použít obousměrný seznam

Polynomy

- v každém prvku LSS uložen koeficient a exponent jednoho členu polynomu, členy s nulovým koeficientem se do LSS neukládají

- vhodné pro „řídké polynomy“ vyšších stupňů

- pro provádění operací je výhodné mít LSS uspořádaný (vzestupně nebo sestupně) podle hodnot exponentu

Druhy lineárních spojových seznamů

- obyčejný seznam *(dosud)*

- obousměrný seznam

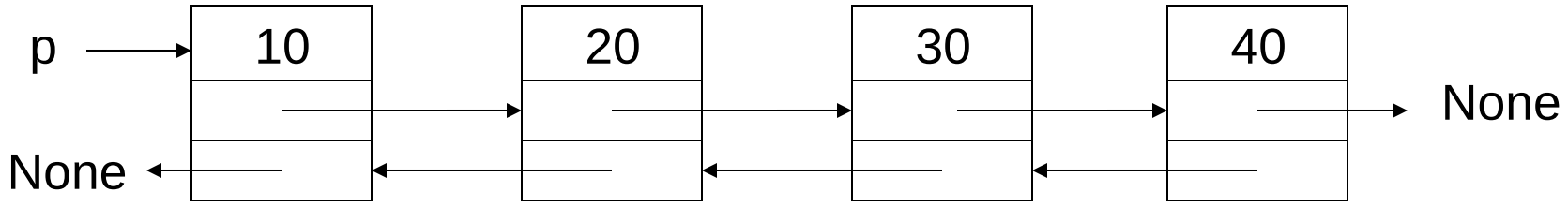
- cyklický seznam

- seznam s hlavou

Obousměrný seznam

```
class Uzel:
    """uzel obousměrného spojového seznamu"""

    def __init__(self, x = None):
        self.info = x                # uložená hodnota
        self.za = None               # následník
        self.pred = None             # předchůdce
```



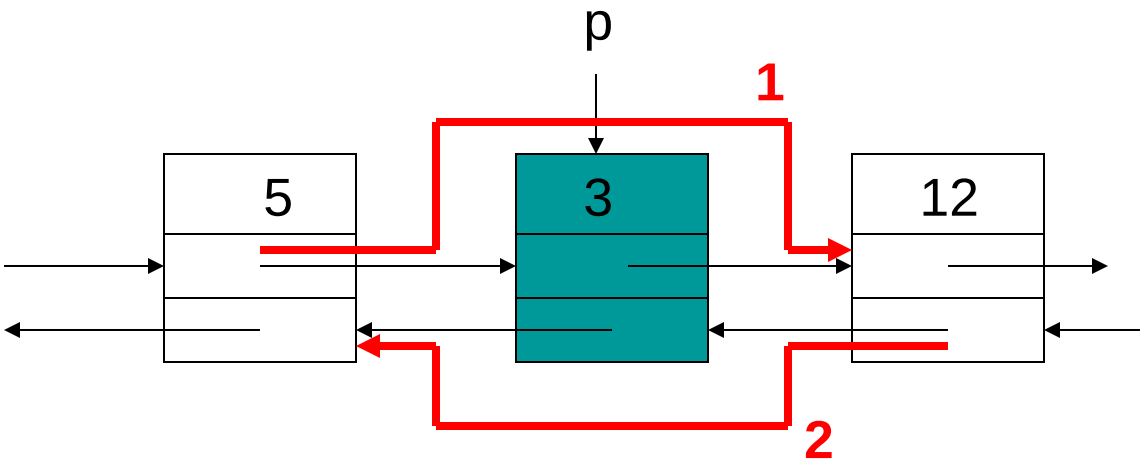
Vlastnosti

- paměťově náročnější než jednosměrný seznam
- umožňuje procházení seznamem střídavě oběma směry
- některé operace obtížnější (přepojuje se více ukazatelů), jiné naopak snadnější

Příklad: vypuštění prvku z obousměrného LSS

- máme ukazatel p na rušený prvek (p není okrajový prvek)

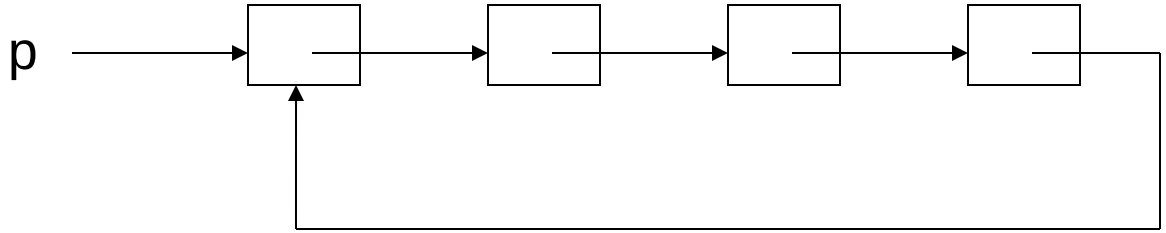
```
p.pred.za = p.za;    {1}
p.za.pred = p.pred;  {2}
```



Cyklický seznam

- poslední prvek seznamu neodkazuje na **`None`**, nýbrž na první prvek

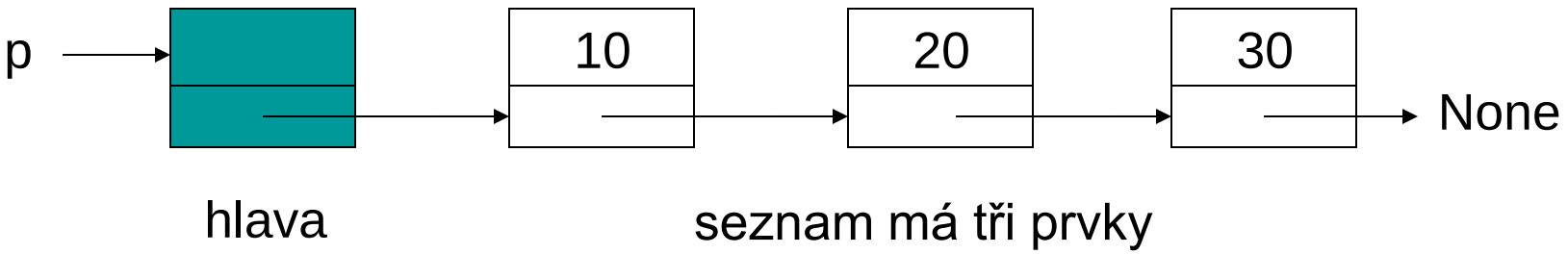
- lze použít i pro obousměrné seznamy, potom navíc první prvek seznamu odkazuje svým ukazatelem `pred` na poslední prvek



Seznam s hlavou

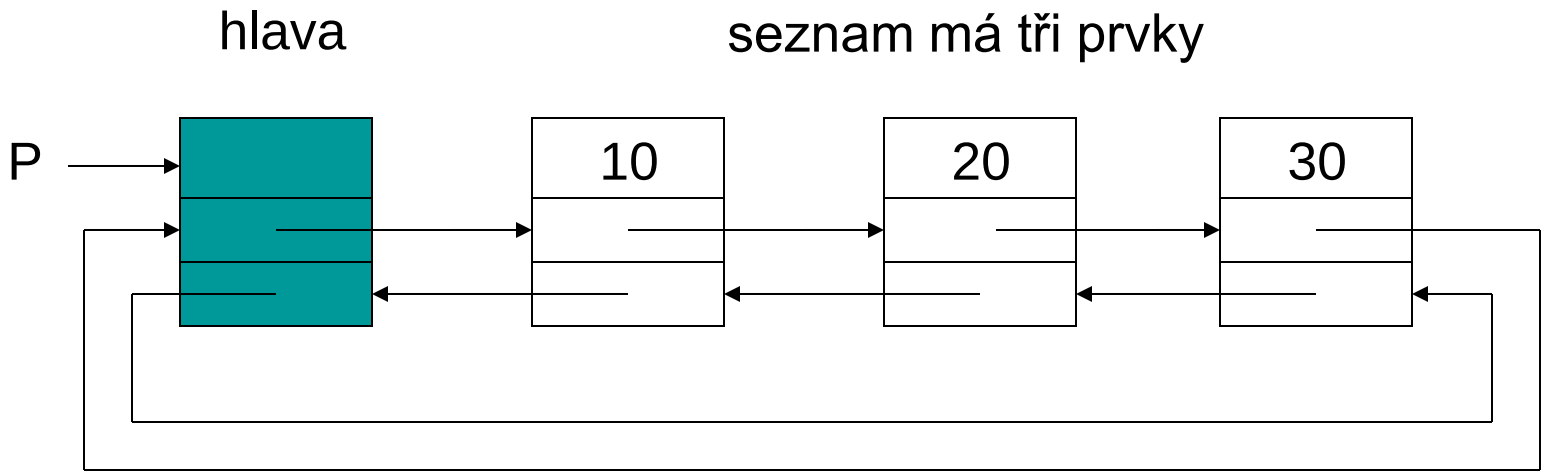
hlava = jeden prvek navíc umístěný na začátku seznamu, neobsahuje uloženou hodnotou
(příp. atribut `info` v hlavě lze využít pro jiné účely)

Použití:
prázdný seznam je tvořen samotnou hlavou (není to tedy jen **`None`** jako u obyčejných seznamů) → snáze se programují některé akce spočívající v přidávání resp. odebírání prvků ze seznamu
(není třeba stále rozlišovat případy, zda seznam je či není prázdný).



Možné kombinace

- např. obousměrný cyklický lineární spojový seznam s hlavou



Dočasná hlava

Někdy pracujeme s obyčejnými spojovými seznamy, ale pro snazší realizaci požadované operace se vyplatí použít u vytvářeného výsledného seznamu dočasnou hlavu a před vrácením výsledku ji opět zrušit.

Příklad:
Uzly zadaného lineárního spojového seznamu celých čísel rozdělit do dvou seznamů takto:
- do jednoho dát sudé hodnoty (se zachování vzájemného pořadí),
- do druhého dát liché hodnoty (se zachování vzájemného pořadí).