matfyz

Computer Graphics Charles University

Kódování rastrových obrázků

CGG MFF UK Praha

pepca@cgg.mff.cuni.cz

https://cgg.mff.cuni.cz/~pepca/

Použití

Úsporné uložení rastrového obrázku

- proti běžným textovým algoritmům lze využít dvojrozměrné povahy dat

Efektivnější operace s jednoduchými obrázky a **bitovými maskami**

- množinové operace s bitovými maskami
- superpozice obrázků

RLE kódování („Run-Length Encoding”)

Využívá se **koherence** ve vodorovném směru

- sousední pixely mají často stejnu hodnotu
- nejvýhodnější u málo barevných obrázků

Speciální příznak pro uložení **„běhu”**

`ESC {počet} {pixel}` (PCX)

Dva typy paketů – **„kopírovací”** a **„opakovací”**

`COPY {počet} {data ...}` (Targa, BMP…)

`FILL {počet} {pixel}`

Kvadrantový strom („quadtree”)

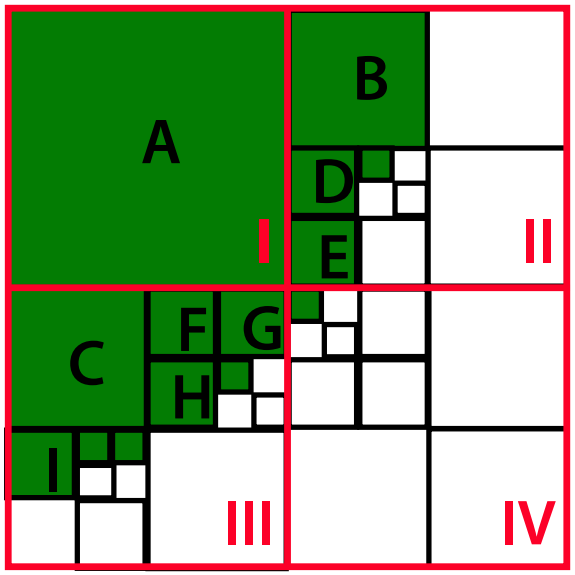
Využívá se **koherence** ve vodorovném i svislém směru

- úsporně se kódují větší souvislé plochy jedné barvy
- **adaptivní princip**
  - » postupné dělení „zajímavých” (=členitých) oblastí

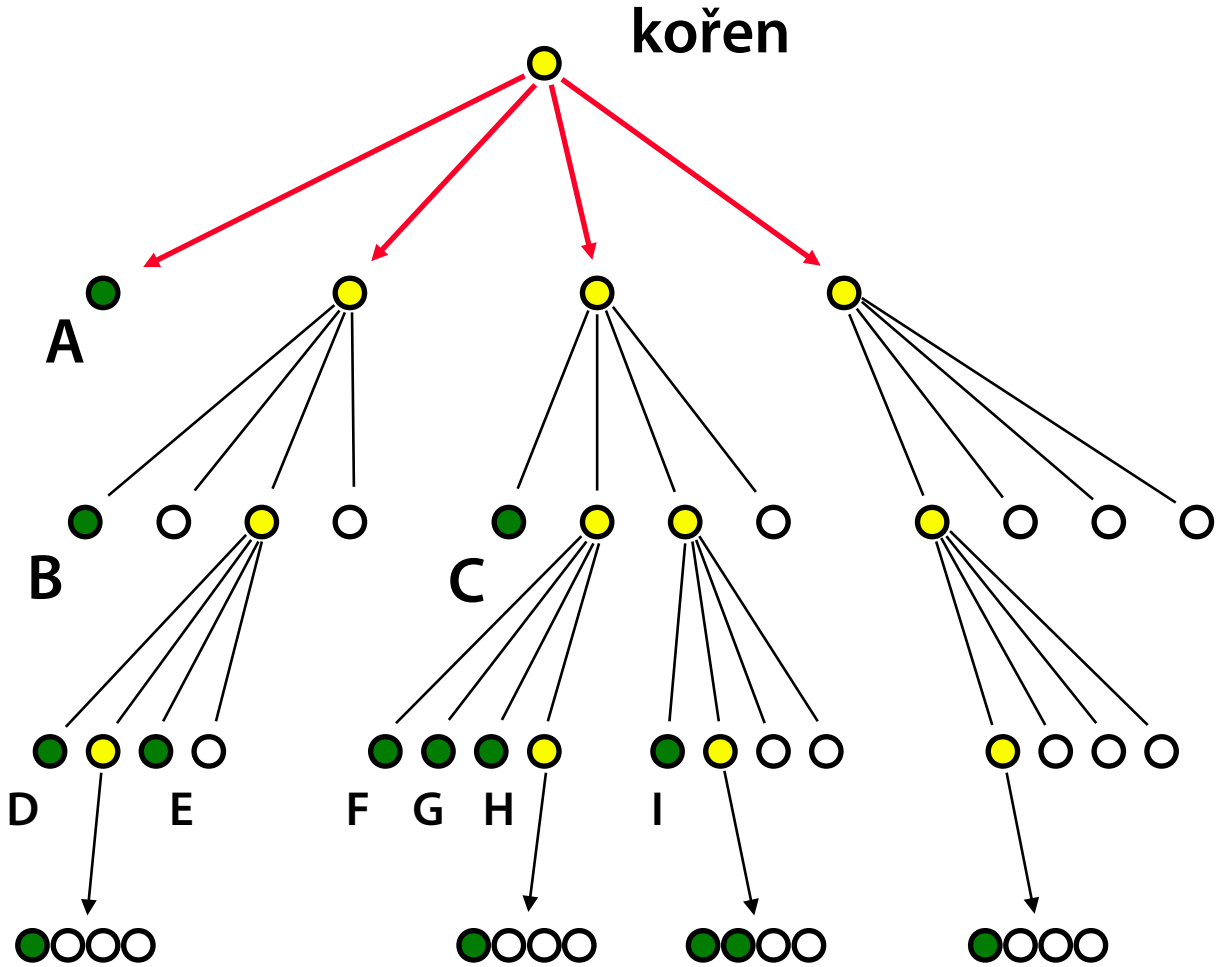
Aplikace kvadrantového stromu

- kódování obrazu
- úsporné uložení **bitové masky** (množinové operace)
- pomocná datová struktura pro **rychlé vyhledávání**

Kvadrantový strom („quadtree”)



16 × 16
(256 bytů)



12 záznamů (96 bytů)

Kódování kvadrantového stromu

Podle definice (metoda „shora-dolů”)

- daný čtverec se zkontroluje ⇒ je-li vícebarevný, rozdělí se na čtyři části, atd. (rekurze, „pre-order”)
- hodnoty některých pixelů se čtou **několikanásobně**

Metoda **„zdola-nahoru”**

- začíná se od **čtverečků 2×2**, jednobarevné oblasti se spojují do větších uzlů grafu, atd. … a nakonec se vytvoří **kořen stromu** (rekurze, „post-order”)
- každý pixel se čte pouze **jedenkrát**

Množinové operace

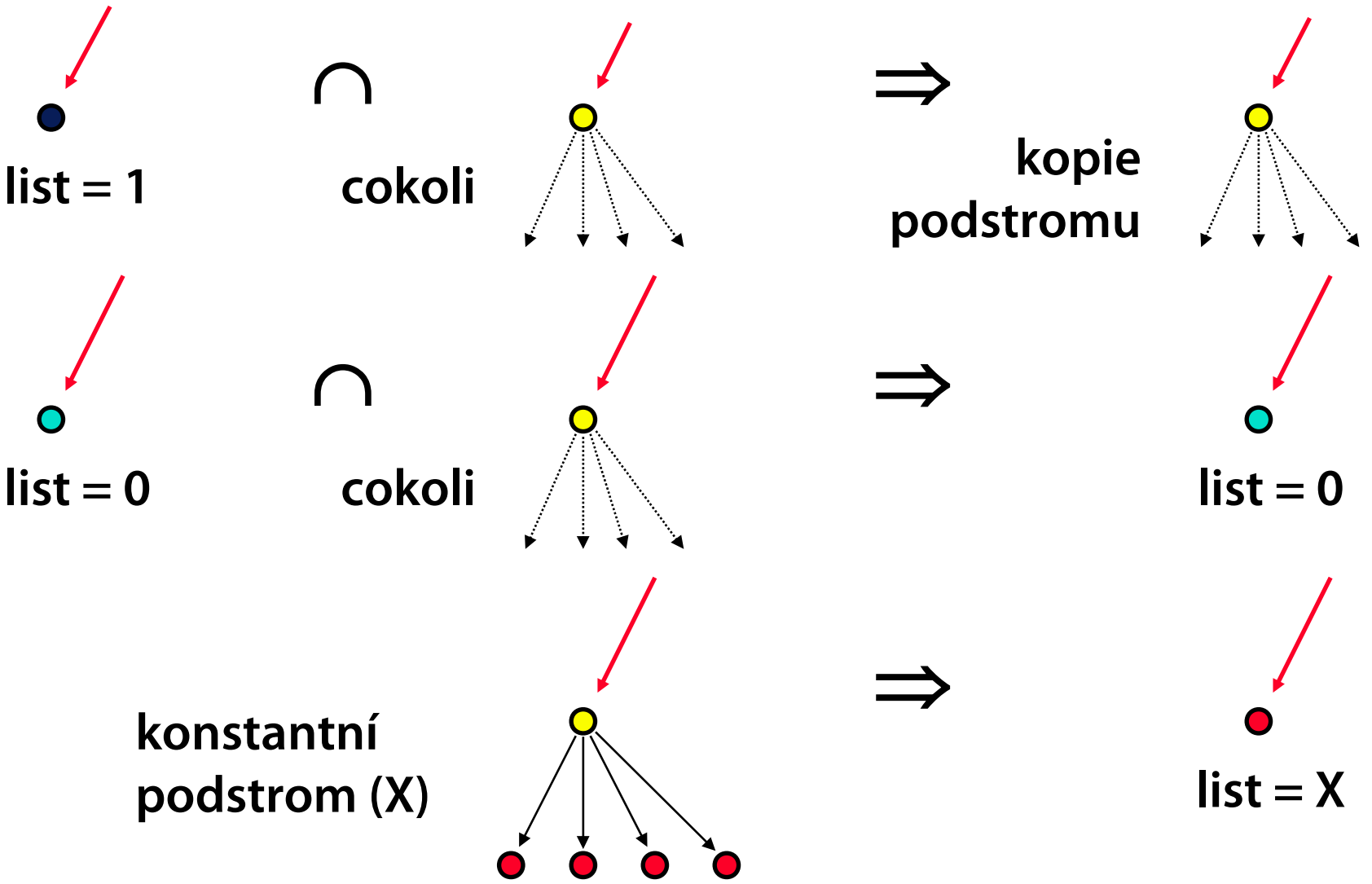
Kvadrantové stromy reprezentují **jednobitovou informaci** (množinu, masku)

- množinové operace (sjednocení, průnik, rozdíl, ..)
- předpokládá se shodný definiční obor operandů

Prochází se paralelně všechny **vstupní stromy** a současně se konstruuje **výsledný strom**

- všechny vstupní uzly jsou vnitřní $\Rightarrow$ „rozděl a panuj”
- jeden vstupní uzel je listem $\Rightarrow$ podle typu množinové operace se zpracují ostatní vstupní podstromy

Příklad – pravidla pro operaci „průnik”



Implementační poznámky

Kódování **obecné oblasti**

- zakóduje se nejmenší čtverec rozměru $\mathbf{2^n \times 2^n}$, který danou oblast obsahuje
- pixely ležící **mimo oblast** se zakódují speciální hodnotou („outside")
- jiná varianta: vnějším pixelům se přiřadí hodnota **okrajových pixelů** („don't care") – největší úspora

Úsporné **hybridní kódování** obrázku

- je-li podstrom větší než bitmapa, ukládám **bitmapu**

Implementační poznámky II

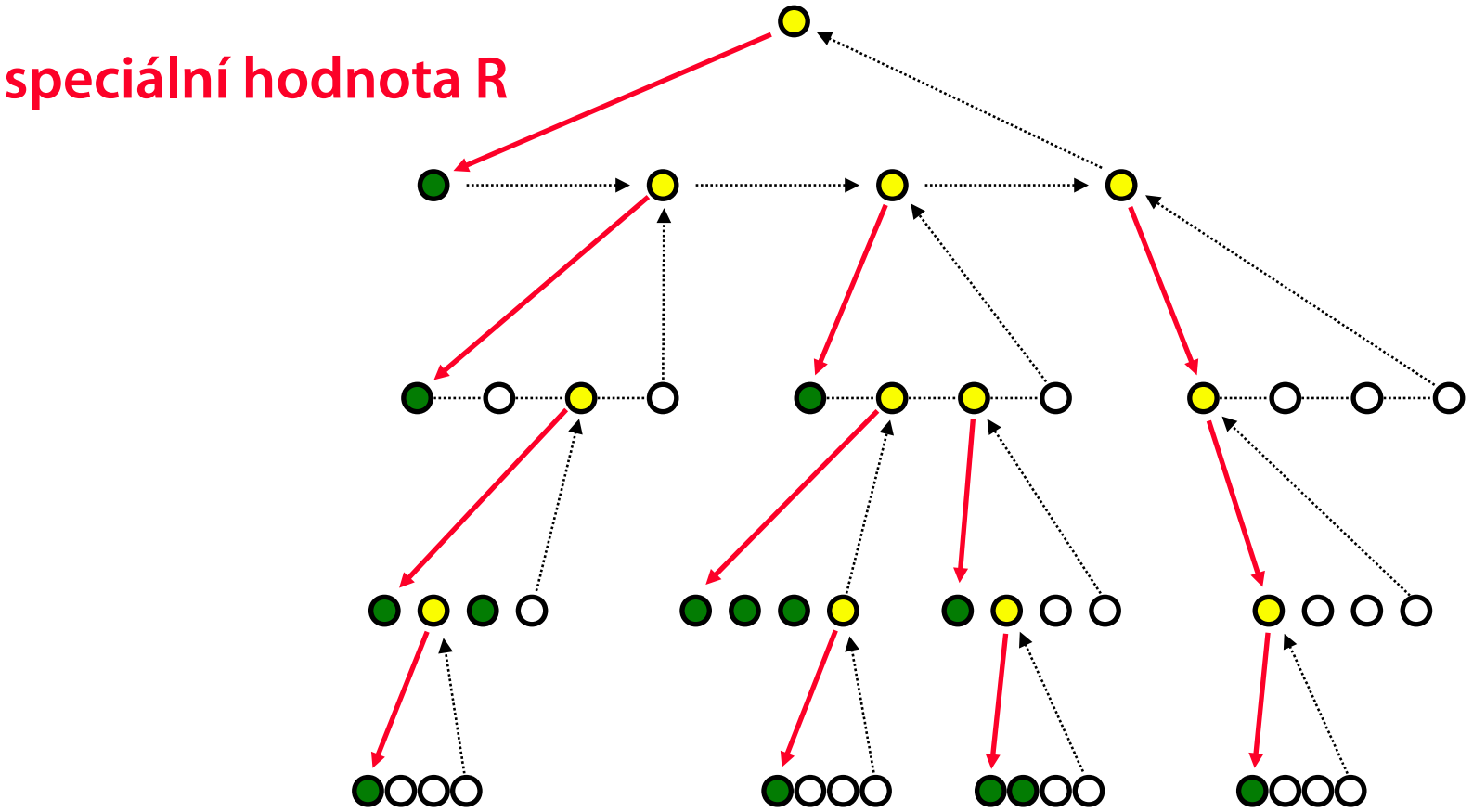
Společné větve kvadrantového stromu

- **opakuje-li** se ve stromu nějaká větev (podstrom) několikrát, uloží se pouze jednou a pak se na ně může odkazovat i odjinud
- ze stromu se stává hierarchický **graf** (ADG – acyklický orientovaný graf)
- společná větev může být použita v různých úrovních

Lineární uložení kvadrantového stromu (serializace)

- **průchod stromem** zleva-doprava („pre-order”)

Lineární uložení



R1R10R1R1000100R1R111R1000R1R1100000
RRR1000000000 … 49 položek

Řádkový seznam („X-transition list”)

Rastrová reprezentace **množiny** (jednobitové masky) **v rovině**

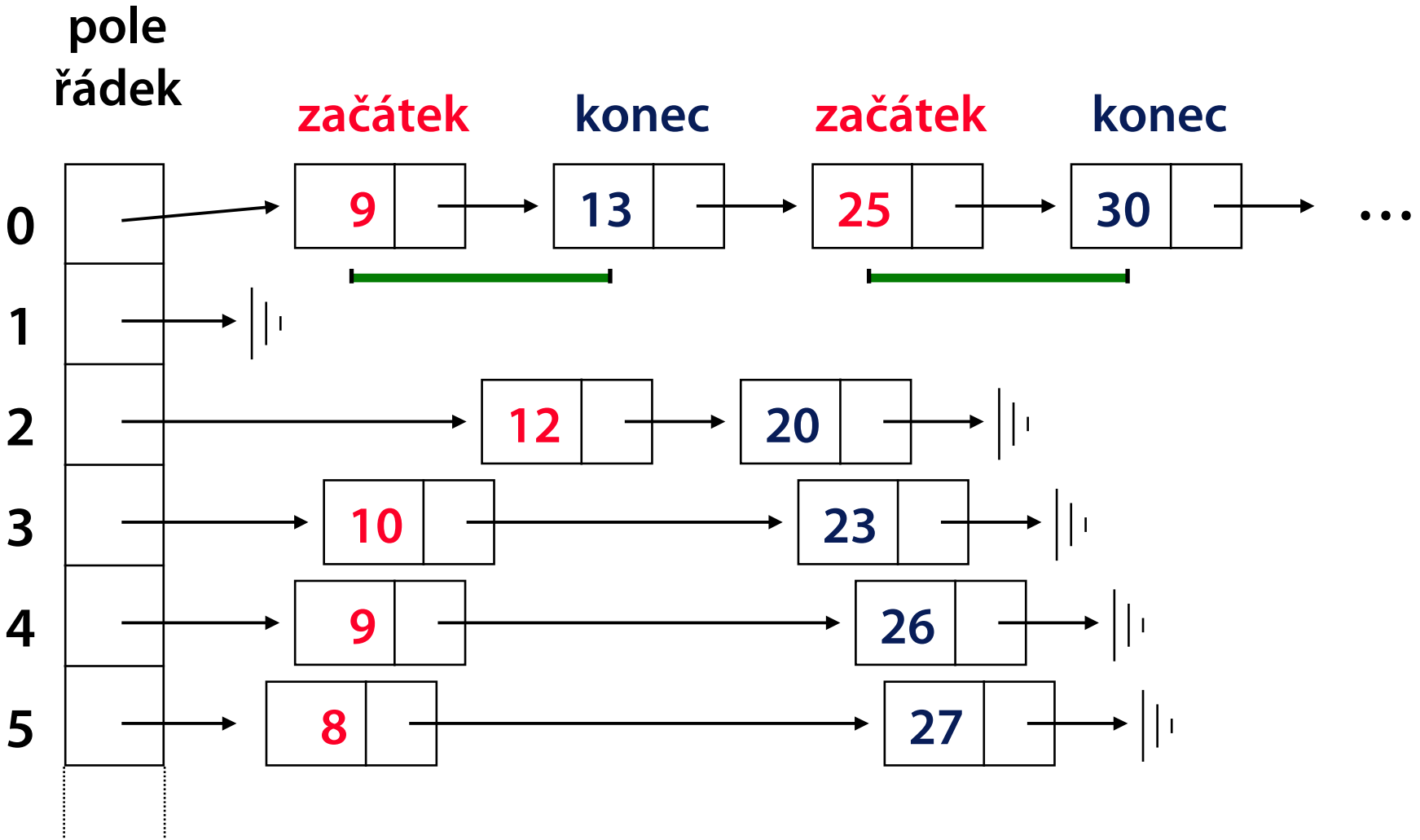
- efektivní implementace **množinových operací** (slévání uspořádaných seznamů)
- lze použít při **vyplňovacích algoritmech**

Výhodná pro **oblasti s jednoduchým okrajem**

Pro každou řádku se ukládá uspořádaný **seznam pixelů**, kterými prochází hranice oblasti

- v těchto pixelech se mění **0** → **1** nebo **1** → **0**

Řádkový seznam změn



Množinové operace

Doplněk

- v každé řádce se přidá/odstraní prvek **[0]**

Binární operace – slévají se seznamy změn příslušných vstupních řádek

- nonekvivalence (**XOR**) je nejsnazší – jen se slévá (a odstraňují duplicitní záznamy)
- u jiných operací se zařazují na výstup jen některé záznamy (podle stavové pomocné proměnné)

Množinová operace na jedné řádce

```
void MergeLists (HList L1, HList L2, out HList Result)
{
  Result.Init();
  bool state = false;              // výstupní stav
  bool in1 = false, in2 = false;   // vstupní stavy
  while (!L1.Empty && !L2.Empty)
  {
    int x = min(L1.First, L2.First);
    if (x == L1.First)             // odebírám z L1
    {
      in1 = !in1; L1.Get();
    }
    if (x == L2.First)             // odebírám z L2
    {
      in2 = !in2; L2.Get();
    }
    if (BooleanOperation(in1, in2) != state)
    {
      Result.Put(x);
      state = !state;
    }
  }
  // ...dočtení zbytku neprázdného vstupního seznamu
}
```

Literatura

J. Foley, A. van Dam, S. Feiner, J. Hughes: Computer Graphics, Principles and Practice, 844-846, 552-555, 992-996