Kódování rastrových obrázků

KSVI MFF UK Praha

e-mail: Josef.Pelikan@mff.cuni.cz

WWW: http://cgg.ms.mff.cuni.cz/~pepca/

Použití

- **úsporné uložení** rastrového obrázku
  - proti běžným textovým algoritmům mohu využít dvojrozměrné povahy dat
- efektivnější operace s jednoduchými obrázky a **bitovými maskami**
  - množinové operace s bitovými maskami
  - superpozice obrázků

RLE kódování ("run-length enc.")

- využívá se **koherence** ve vodorovném směru:
  - sousední pixely mají často stejnu hodnotu
  - nejvýhodnější u málo barevných obrázků

- **speciální (pří)znak** pro uložení **"běhu"**:

  ESC {počet} {pixel} (PCX)

- dva typy paketů - **"kopírovací"** a **"opakovací"**:

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

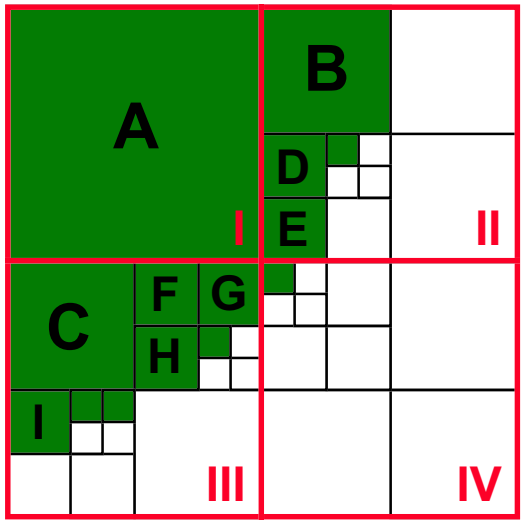
  FILL {počet} {pixel}

Kvadrantový strom (“quadtree”)

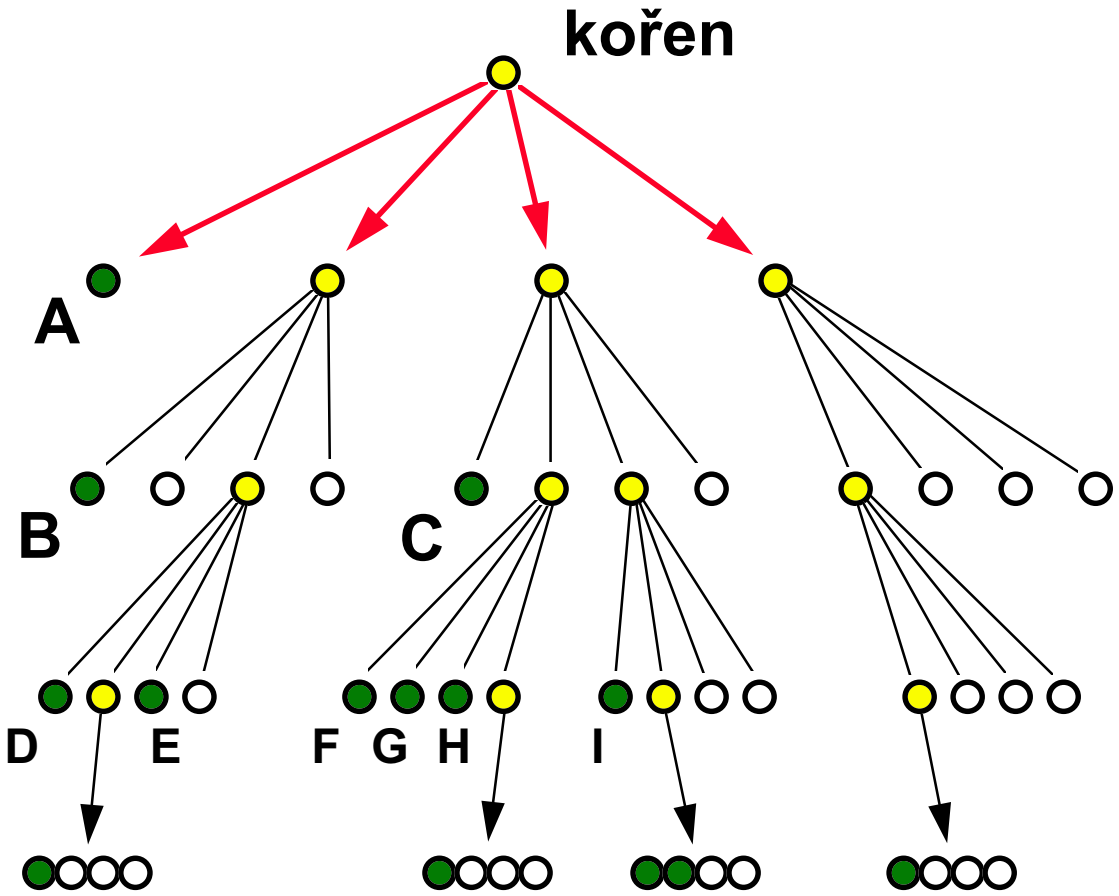
- využívá se **koherence** ve vodorovném i svislém směru
  - úsporně se kódují větší souvislé plochy jedné barvy
  - **adaptivní princip** (postupné dělení “zajímavých”= členitých oblastí)

- **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)

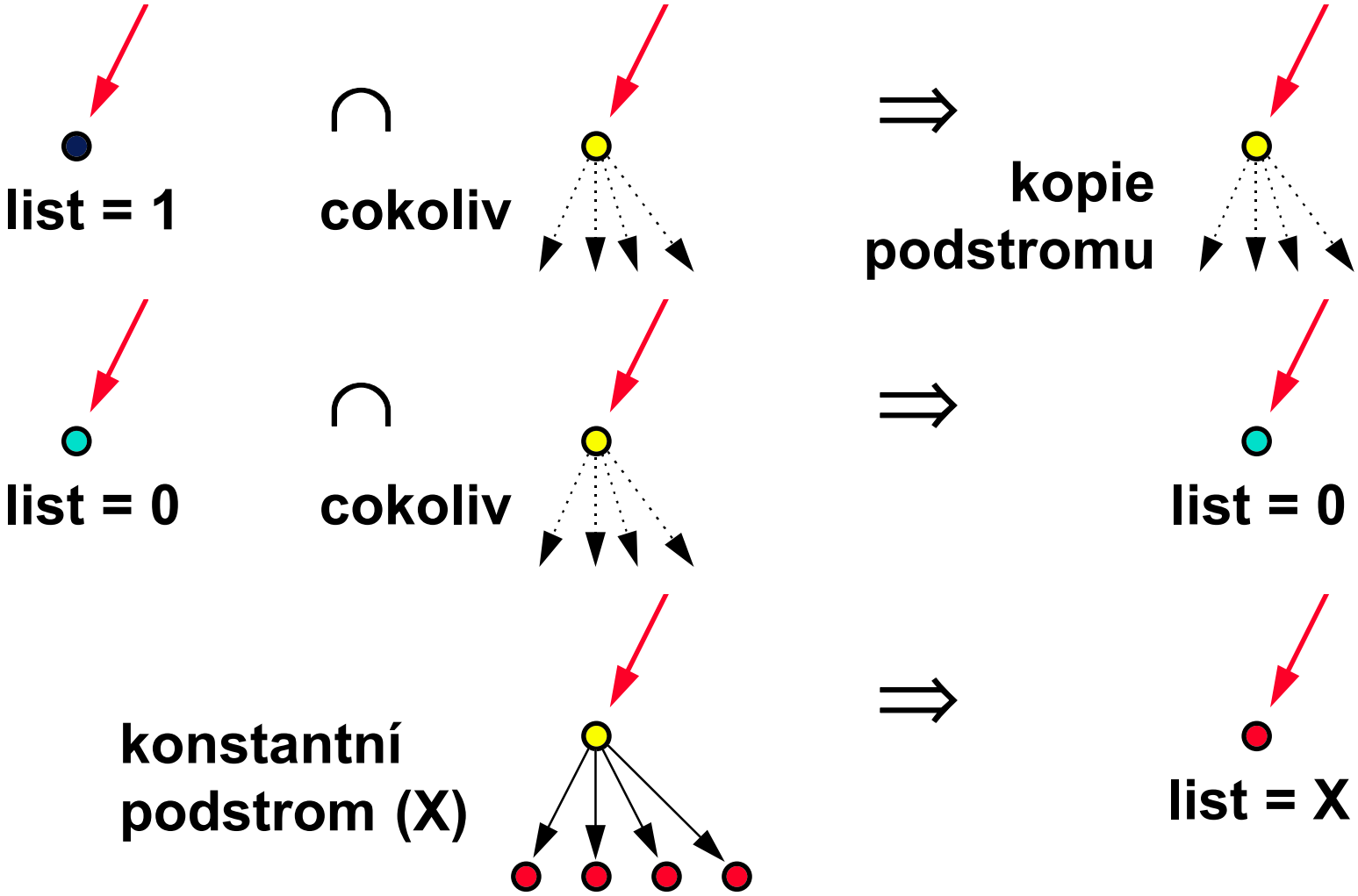
Kódování kvadrantového stromu

- **podle definice** (metoda “shora-dolů”):
  - daný čtverec zkontroluji ⇒ je-li vícebarevný, rozdělím ho na čtyři části, atd. (rekurze, “pre-order”)
  - hodnoty některých pixelů čtu **několikanásobně**
- metoda **“zdola-nahoru”**:
  - začínám od **čtverečků 2×2**, jednobarevné oblasti spojuji do větších uzlů grafu, atd., ... nakonec vytvořím **kořen stromu** (rekurze, “post-order”)
  - každý pixel čtu pouze **jedenkrát**

Množinové operace

- kvadrantové stromy reprezentují **jednobitovou informaci** (množinu, masku)
  - množinové operace (sjednocení, průnik, rozdíl, ..)
  - předpokládám shodný definiční obor operandů
- procházím paralelně všechny **vstupní stromy** a současně konstruuji **výsledný strom**:
  - všechny vstupní uzly jsou vnitřní ⇒ “rozděl a panuj”
  - jeden vstupní uzel je listem ⇒ podle typu množinové operace zpracuji ostatní vstupní podstromy

Pravidla pro operaci “průnik”:



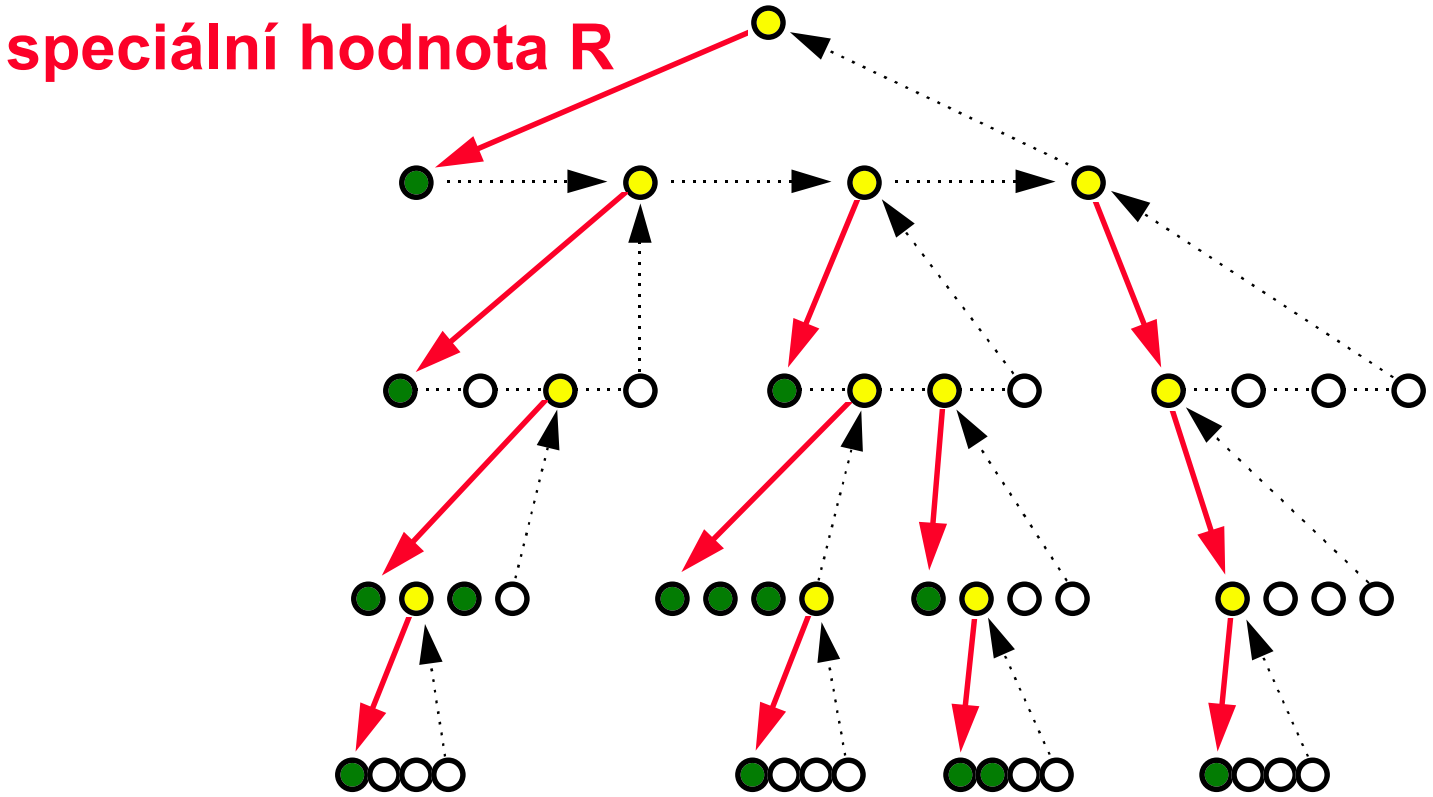
Implementační poznámky

- kódování **obecné oblasti**:
  - zakóduji nejmenší čtverec rozměru $\mathbf{2^n \times 2^n}$, který danou oblast obsahuje
  - pixely ležící **mimo oblast** zakóduji speciální hodnotou (“outside”)
  - jiná varianta: vnějším pixelům přiřadím hodnotu **okrajových pixelů** (“don’t care”) - největší úspora
- úsporné **hybridní kódování** obrázku:
  - je-li podstrom větší než bitmapa, ukládám **bitmapu**

Implementační poznámky

- **společné větve** kvadrantového stromu:
  - **opakuje-li** se ve stromu nějaká větev (podstrom) několikrát, uložím ji pouze jednou a odkazuji se na ni z více míst
  - ze stromu se stává hierarchický **graf** (ADG - acyklický orientovaný graf)
  - společná větev může být použita v různých úrovních
- **lineární uložení** kvadrantového stromu (disk)
  - **průchod stromem** zleva-doprava (“pre-order”)

Lineární uložení kvadr. stromu



R1R10R1R1000100R1R111R1000R1R1100000
RRR1000000000 **... 49 položek**

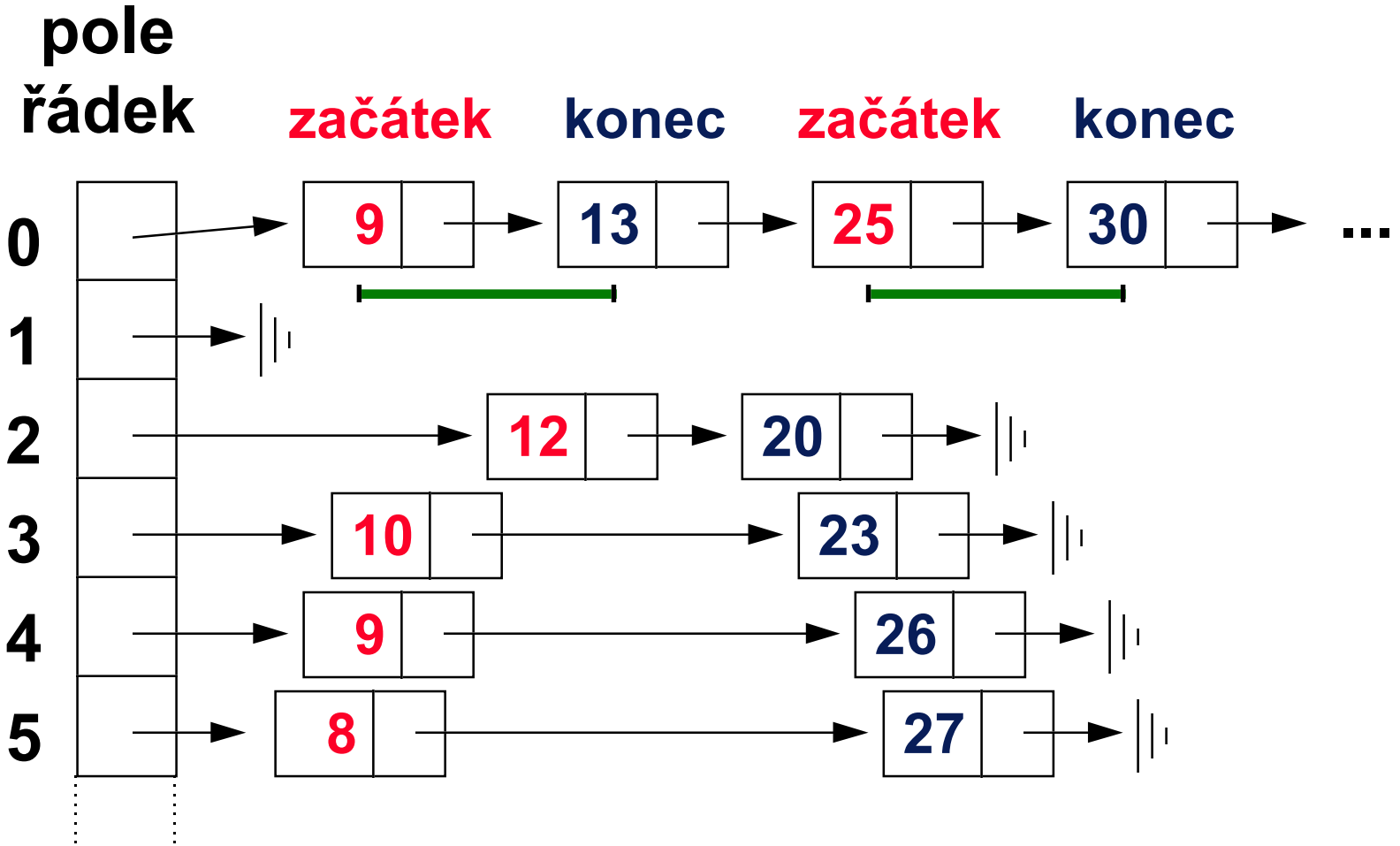
Řádkový seznam změn
("X-transition list")

- 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 ukládám 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 přidám/odstraním prvek **[0]**

- binární operace - **slévám seznamy změn** příslušných vstupních řádek:
  - nonekvivalence (**XOR**) je nejsnazší - dělám jenom slévání (a odstraňuji duplicitní záznamy)
  - u jiných operací zařazuji na výstup jen některé záznamy (stavové pomocné proměnné)

Množinová operace na 1 řádce

```
procedure MergeLists ( var L1, L2, Result : list );
var state, newstate, in1, in2 : boolean;
    x : integer;
begin
  Result.Init; state := false;        { výstupní seznam }
  in1 := false; in2 := false;         { vstupní stavy }
  while (not L1.Empty) and (not L2.Empty) do begin
      x := minimum(L1.First,L2.First);
      if x = L1.First then begin      { odebírám z L1 }
          in1 := not in1; L1.Get; end;
      if x = L2.First then begin      { odebírám z L2 }
          in2 := not in2; L2.Get; end;
      newstate := BooleanOperation(in1,in2);
      if newstate <> state then Result.Put(x);
      state := newstate;
    end;
  { dočtení zbytku vstupního seznamu }
end;
```

Konec

Další informace:

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

- **LAN** na Malé Straně:
  - **barbora\usr:\vyuka\pelikan\5**