Interaktivna vyuka kompresie dat

4. januára 2003

Zakladne udaje

Meno a priezvisko	Peter Hudec (hudecof@hudecof.sk)
Diplomovy veduci	doc. Jaroslav Polec, FEI KTL, Slovensko

Moja predstava

Applet bude ukazovat iba zakladne informacie o priebehu algoritmu. Implicitne sa bude predpokladat, ze uzivatel si precital popis algoritmu, kde bude vysvetleny aj s pripadnymi konstantami [1]. Cinnost appletu sa da rozdelit na niekolko casti

- nastavenie premennych (retazec, konstanty)
- vyber kompresie alebo dekompresie daneho retazca
- samotna vyuka: step, run, reset

Nastavenie premennych spociva vo vybrati kodovaneho slova, alebo postupnosti bitov. [2]. Okrem preddefinovanych vstupov, je moznost zadanie aj vlastneho vstupu[3] Pokial algoritmus obasahuje aj nastavenie, ktore dokazu je ucinnost rapidne zmenit *buffer size - LZ77, window size - RLE*

Vyber kompresie urcuje, ktory smer sa chce uzivatel naucit. Pri **kompresii** sa bude vysledok komprimovany text v citatelnej forme. Pri **dekompresii** bude za vstupne slovo povazovane komprimovane vstupne slovo a vysledkom bude slovo zadane uzivatelom.

Interaktivita samotnej vyuky je zabezpecena troma tlacitkmai s nazvami *Step*, *Run* a *Reset*. Myslim, ze nazvy su samovysvetlujuce.

Realizacia

Ako uz bolo spomenute, na realizaciu interaktivity je pouzite Java vo verzii 1.3.1[4].

Pri realizacii som vyrobil niekolko tried, ktore mi neskor ulahcia vytvarenie algoritmov, alebo samotheho uzivatelskeho rozhrania predefinovanim minimalneho poctu medot. Medzi ne patria triedy *MyJComponent, Tracer, ConfigurationPanel, ControlButtons, ...*

Za vsetky spomeniem iba komponentu **MyJComponent**. Je potomkom triedy *JComponent*, co je zakladna trieda na tvorbu vizualnych kompenentov v jazyku Java. Jej vyhodou je pridanie jedneho noveho Listeneru [5], s nazvom *DrawListener*.

[1] zobrazovane appletom, alebo nastavitelne
[2] vo velkej miere zavisi od algoritmu a jeho varianty
[3] tato moznost v papierovej forme chyba
[4] najnovsia verzia je JDK 1.4
[5] nieco podobne ako event vo Win

```
public interface DrawListener extends EventListener {
  public void actionDrawBefore(DrawEvent e);
  public void actionDrawAfter(DrawEvent e);
}
```

V tejto komponente je predefinovana funcia *draw*, ktora najprv zavola metody, ktore kreslia pred jej samotnym vykresenim, potom sa vykresli a zavola funkcie, ktore kreslia po jej vykresleni. Takze v jej nasledovnikoch staci predefinovat funkciu *innerDraw*, ktora zabezpecuje samotne vykreslenie komponenty. Zivim prikladom je komponenta **ShowString**.

Uz sa stalo skutocnostou

V sucastnosti je uz plne implementovany bitovy RLE algoritmus. Bohuzial, este k nemu neexistuje plna vizualizacia, takze uzivatal musi vediet ako dany algoritmus pracuje.

Moje vyuka bude aj pocas vyvoja online zverejnena na stranke: `http://www.hudecof.sk/projekty/diplomovka/`

□

Obr. 1: Ciastocne vizualizovane RLE