Adaptívne huffmanove kódovanie
Dokumentácia k projektu č. 1

Peter Lacko
xlacko06@stud.fit.vutbr.cz

14. března 2015

Činnosť programu

Program `ahed` slúži na kódovanie a dekódovanie dát s využitím adaptívneho huffmanovho kódovania (AHC). Pracuje nad 8 bitovými symbolmi, pričom veľkosť spracovávaných dát nie je obmedzená. Algoritmus AHC podobne ako obyčajné huffmanovo kódovanie (RHC) ukladá symboly do stromovej štruktúry tak, že symboly s najvyššou frekvencoiu výskytu sa nachádzajú blízko koreňa a zriedkavé symboly ďalej od koreňa. AHC však narozdiel od RHC nepotrebuje dopredu vedieť početnosť symbolov, keďže strom sa vytvára a upravuje počas behu programu. Princíp činnosti je uvedený v Algoritme 1.

Dekódovanie súboru prebieha podobne ako kódovanie, s tým rozdielom že operácie sa vykonávajú zrkadlovo (napr. pri kódovaní pri prvom načítaní znaku zapíšeme najprv kód uzlu ZERO a následne znak zatiaľ čo pri dekódovaní po prečítaní kódu ZERO nasleduje načítanie nového znaku).

Ovládanie aplikácie

Pre spustenie je aplikáciu najprv nutné preložiť príkazom `make` bez parametrov. `ahed` akceptuje nasledujúce parametre:
-h zobrazí nápovedu a ukončí sa,
-c kódovanie vstupných dát,
-x dekódovanie vstupných dát,
-i názov vstupného súboru; ak nezadaný číta sa zo štandardného vstupu,
-o názov výstupného súboru; ak nezadaný zapisuje sa na štandardný výstup,
-l názov logovacieho súboru; ak nezadaný, výstup sa ignoruje.

Príklad kódovania súboru `test.txt`:

```
$ ahed -c -i test.txt -o out -l report && cat report
login = xlacko06
uncodedSize = 768771
codedSize = 438512
```

Algoritmus 1: Adaptívne huffmanovo kódovanie

```
input  : Nekódovaný vstupný súbor
output: Kódovaný výstupný súbor

ahed_encoding()
begin
    vytvor uzol ZERO – počiatočný koreň stromu
    X = nasledujúci vstupný symbol
    while X != koniec súboru do
        if X sa nenachádza v strome then
            zapíš Code(ZERO) na výstup                /* kód uzlu ZERO */
            zapíš X na výstup
            vlož X do stromu
            aktualizuj_strom(Uzol(X))            /* uzol obsahujúci symbol X */
        else
            zapíš Code(X) na výstup
            aktualizuj_strom(Uzol(X))
        end
        X = nasledujúci vstupný symbol
    end
end
zapíš Code(ZERO) na výstup          /* slúži ako zarážka pri dekódovaní */

aktualizuj_strom(Uzol U)
begin
    aktUzol = U
    while aktUzol != koreň do
        if Existuje uzol N s nižším stupňom a rovnakou frekvenciou výskytuako má aktUzol then
            vymeň aktUzol a uzol N spolu s ich podstromami
        end
        inkrementuj frekvenciu uzlu aktUzol
        aktUzol = predchodca aktUzol
    end
    inkrementuj frekvenciu koreňa
    if frekvencia koreňa = maximálna hodnota then              /* pretečenie čítača */
        aktUzol = zmeň_mierku_stromu() /* vráti najľavejší uzol najnižšej úrovne */
        aktualizuj_strom(aktUzol)
    end
    aktualizuj_kódy_uzlov()
end
```