Sada: 3 Příklad: 2 IV003 Algoritmy a datové struktury II

Jméno: Jiří Novotný UČO: 409963

Jméno: Henrich Lauko UČO: 410438

Na efektívne riešenie sme použili dynamické programovanie s technikou memoizácie predchádzajúcich využitých sekvencií slov. Pri memoizácii využívame maticu veľkosti $(|z|+1) \times (|z|+1)$. Kde na pozíci $matrix[i][j]$ držíme pravdivostú hodnotu, či je možné v i krokoch cez slovo x a j krokoch cez slovo y prejsť $i+j$ znakov zo slova z. Teda pozícia $matrix[0][0]$ nám vyjadruje, spojenie prázdneho slova, ktoré dokážeme vytvoriť z ľubovoľných slov mocnených na 0 ($x^0 = \epsilon$), takže bude vždy platné. V algoritme sú slová a matica indexované od 0, preto indexy obsahujú -1. Index x_0 teda označuje prvý znak slova x. Pri indexácii slov využívame zvyšky po delení dĺžkou daných slov pre prípady repetície(pozícia v slove je definovaná ako $x_{i-1 \bmod |x|}$). Môžeme si slová x a y predstaviť aj ako repetíciu sama seba veľkosti slova z. Slovo $z$považujeme za zreťazenie v prípade, že nejaké miesto na diagonále matice je True, čo znamená, že existuje postúpnosť znakov zo slov x a y tvoriace prechod maticou a tým pádom aj spojenie z.

Procedura isConnection(x, y, z)

vstup: slovo z, ktorému overujeme, či je spojením slov x a y
výstup: rozhodnutie, či z je spojenín x a y

```
// inicializaciá memoizačnej matice
matrix ← matica veľkosti (|z| + 1) × (|z| + 1)
matrix[0][0] ← True
// inicializácia kraju matice
for i ← 1 to |z| do
    matrix[0][i] ← (matrix[0][i − 1] ∧ x_{i−1 mod |x|} = z_{i−1}) // prechod pod slovom x
    matrix[i][0] ← (matrix[i − 1][0] ∧ y_{i−1 mod |y|} = z_{i−1}) // prechod pod slovom y
od
if matrix[0][|z|] ∨ matrix[|z|][0] then
    return True // z je mocninou len jedného slova
fi
// memoizácia
for i ← 1 to |z| do
    for j ← 1 to |z| − i do
        if (matrix[i][j − 1] ∧ x_{j−1 mod |x|} = z_{i+j−1}) then
            matrix[i][j] ← True // existuje prechod pod znakom zo slova x
        else if (matrix[i − 1][j] ∧ y_{i−1 mod |y|} = z_{i+j−1}) then
            matrix[i][j] ← True // existuje prechod pod znakom zo slova y
        else
            matrix[i][j] ← False // neexistuje prechod pod žiadnym znakom
    od
    if matrix[i][|z| − i] then
        return True // dokázali sme nájsť prechod znakov až ku koncu slova z
    fi
od
return False // slovo z není spojením slov x a y
```

Sada: 3 Příklad: 2 IV003 Algoritmy a datové struktury II

Jméno: Jiří Novotný UČO: 409963

Jméno: Henrich Lauko UČO: 410438

Časová zložitosť: Algoritmus počas svojho výpočtu vypĺňa maticu veľkosti $(|z|+1) \times (|z|+1)$, kde cez každú pozíciu iteruje práve jeden krát. Vzľadom na konštatné operácie vnútri cyklov je časová zložitošt prvého cyklu $z \in \mathcal{O}(z)$ a druhých dvoch zanorených cyklov $\mathcal{O}(z^2)$, teda celková zložitosť je $(z \cdot z/2) \in \mathcal{O}(z^2)$.

Konečnosť: V cykloch algoritmu rastú indexy vždy o 1, vzhľadom k tomu, že indexy v tele cyklu nemeníme z vlastností for-cyklu vieme, že algoritmus,vykoná v prvom cykle len z iterácií v druhých dvoch zanorených cykloch práve $z \cdot z/2$, teda aj skončí.

Korektnosť: Algoritmus je korektný, ak je konečný a parciálne korektný. Konečnosť sme už ukázali a parciálnu korektnosť môžeme dokázať indukciou. Môžeme tvrdiť, že pre dĺžku slova $|z| = 0$ algoritmus platí (jak sme už v úvode rozobrali ϵ je vždy spojením nejakých dvoch slov x a y). Teda predpokladajme, že pre ľubovolné slovo dĺžky n algoritmus vráti korektný výsledok. Chceme dokázať, že ak toto slovo predĺžime o ľubovolný znak algoritmus bude stále korektný. Z indukčného predpokladu vieme, že máme maticu vyplnenú správne pod diagonálov. Pre doplnenie hodnôt na diagonále využívame vedomostí predchádzajúcich krokov z matice a aktuálnu pozíciu v slovách x a y. Korektnosť správneho výsledku na pozícii $matrix[i][j]$, kde $i = j$ nám zaisťujú podmienky na riadkoch 12 - 17, kde vychádzame z už správnych výsledkov a posunu buď po znaku zo slova x alebo y, ak takýto predhod neexistuje, tak ani spojenie nemôže existovať. Teda je algoritmus korektný.

Poznámka: Pre memoizáciu by bolo možné použiť aj 3-dimenzionálne pole veľkosti $x \times y \times z$. Čo by v prípade veľkého z ($z > x \cdot y$) bolo efektívnejšie riešeni. Pre jednoduchší zápis a prehľadnosť sme ale zvolili riešenie pomocou matice.