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

Jméno: Karel Kubíček UČO: 408351

Jméno: Martin Hanžl UČO: 410497

Popis algoritmu: pro konstrukci tohoto algoritmu jsme využili principu dynamického programování. Konkrétně místo opakovaného počítání výsledků v rekurzi si mezivýsledky pamatujeme pomocí memoizace. K tomu nám složí matice rozměrů $(|z|+1) \times (|z|+1)$, na pozici $[i, j]$ uchovává boolovskou hodnotu, vyjadřující, zdali se z i znaků repretice x a j symbolů repetice y dá vytvořit spojení x a y.
Matice je schodovitá, výstup funkce záleží na hodnotách na diagonále, což jsou body se vzdáleností odpovídající celému slovu z, kde cesta maticí popisuje, jak spojení vypadá.
Matici číslujeme od 0, zatímco pozici ve slovech od -1.

Procedura isConnection(x, y, z)

```
   vstup: o slovu z, zjišťujeme, zdali je spojením slov x a y
   výstup: boolovská hodnota, zdali z je spojením x a y
   // matice pro memoizaci
 1 matrix ← matice (|z| + 1) × (|z| + 1)
 2 matrix[0][0] ← True // prázdné slovo z x a y umíme vytvořit
   // inicializace okraje matice, tedy pro slova, které jsou jen repeticí jednoho ze slov
 3 for i ← 1 to |z| do
 4     matrix[0][i] ← (matrix[0][i − 1] ∧ x_{i−1 mod |x|} = z_{i−1}) // přechod pod slovem x
 5     matrix[i][0] ← (matrix[i − 1][0] ∧ y_{i−1 mod |y|} = z_{i−1}) // přechod pod slovem y
 6 od
 7 if matrix[0][|z|] ∨ matrix[|z|][0] then
 8     return True // z je repeticí jen jednoho slova
 9 fi
   // memoizace
10 for i ← 1 to |z| do
11     for j ← 1 to |z| − i do
12         if (matrix[i][j − 1] ∧ x_{j−1 mod |x|} = z_{i+j−1}) then
13             matrix[i][j] ← True // existuje přechod pod aktuálním znakem ze slova x
14         else if (matrix[i − 1][j] ∧ y_{i−1 mod |y|} = z_{i+j−1}) then
15             matrix[i][j] ← True // existuje přechod pod aktuálním znakem ze slova y
16         else
17             matrix[i][j] ← False // neexistuje přechod pod žádným znakem
18     od
19     if matrix[i][|z| − i] then
20         return True
21 od
22 return False
```

Asymptotická časová složitost: inicializace matice $(|z|+1) \times (|z|+1)$ je v $\mathcal{O}|z|^2$. První for cyklus prochází znovu délku slova z. Až 2 zanořené for cykly od řádku 10 jsou zase v kvadratickém čase. Celkově je tedy složitost kvadratická.

Korektnost: Algoritmus je korektní, pokud na vstupech splňujících vstupní podmínku skončí a zároveň je parciálně korektní.

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

Jméno: Karel Kubíček UČO: 408351

Jméno: Martin Hanžl UČO: 410497

Konečnost algoritmu je implikována použitím pouze for cyklů, které iterují nad délkou slova z. Jelikož se proměnné těchto for cyklů mění jen ve forcyklech a to vždy vzhůru, algoritmus se nemá kde zacyklit a skončí.
Parciální korektnost dokážeme indukcí přes obsah matice.
Pro indexy $[0, 0]$ je zřejmé, že prázdné slovo z x a y poskládat lze. Očekávejme, že algoritmus platí pro slovo délky k, pak algoritmus platí i pro slovo délky $k + 1$. Z indukčního předpokladu víme, že máme napočítanou matici po diagonálu ve vzdálenosti k od $[0, 0]$. Na této diagonále jsou informace o možnosti složit slovo délky k. Tuto matici můžeme výpočtem doplnit o novou diagonálu ve vzdálenosti $k + 1$. Pozici v nové diagonále určíme podle levé a dolní pozice. V případě, že jsou obě False, pak ani nová pozice nemůže být True, což odpovídá tomu, že jakmile nejsme schopni spojit nějaká slova, pak nejsme schopni spojit ani jejich slova prodloužené o nějaký symbol. V případě alespoň jednoho z levého, nebo dolního souseda s hodnotou True chceme do slova vzniklého spojením přidat znak, který odpovídá n-tému znaku v repetici slova, ze kterého vybíráme. To je dáno pohybem v matici, pohyb doprava znamená výběr znaku ze repetice slova x a obdobně pro y. Pokud další přidaný znak odpovídá $k + 1$. symbolu slova z, pak přiřadíme na zkoumanou pozici v matici hodnotu True. Algoritmus je tedy korektní