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

Jméno: Karel Kubíček UČO: 408351

Jméno: Henrich Lauko UČO: 410438

Algoritmus pro zpřehlednění pracuje s globálním polem *Array*, které má délku zadaného textu, kterou si také pamatujeme v proměnné n. Pole z počátku obsahuje hodnoty Nil a je indexováno od 1 do n.

Procedura Words(*string*)

vstup: část vstupního textu *string*, při prvním volání odpovídá délce n
výstup: boolovská hodnota, zdali lze vstup rozdělit mezerami tak, aby byl složen ze slov zadaného slovníku

```
1 if Dict (string) then
2     return True // celý string je slovo
3 for i = n downto n − |string| do
4     if Array[i] = Nil then
5         Array[i] ← Words (Array[i, n]) // hodnotu jsem ještě nepočítal, v rekurzi ji
              spočítám poprvé a uložím
6     if Dict (string[0, i − 1]) ∧ Array[i] = True then
7         return True // string[0, i − 1] je ve slovníku a zbytek lze také rozdělit, což víme
              z rekurze
8 od
9 return False // string nelze rozložit
```

Popis algoritmu: tento rekurzivní algoritmus využívá dynamického programování k řešení problému s exponenciálním množstvím podproblému pomocí memoizace. Algoritmus prochází zadaný řetězec odzadu, začneme posledním písmenem, pak dvěma až po celé slovo. Na tomto podřetězci se volá rekurzivně algoritmus znovu, abychom se vyhnuli exponenciálnímu počtu volání, tak si již vyřešené podproblémy pamatujeme v pomocném poli. Procedura vrací návratovou hodnotu True v případě, že rozdělený řetězec v první části obsahuje slovo a zbytek je podle dat získaných z rekurze a uložených v poli také rozdělitelný na slova. Pokud toto nesplňuje a ani celý řetězec není slovem, pak se vrací False, což je buďto použito pro uložení v poli, nebo v případě prvního volání jako návratová hodnota celé procedury.

Korektnost: algoritmus je korektní, pokud je parciálně korektní a pro všechny vstupy splňující vstupní podmínku skončí. Vstupní podmínku nemáme zadáním definovanou (pokud nějakou používá funkce Dict, pak ji náš algoritmus musí splňovat též). Parciální korektností rozumíme, že výstup splňuje výstupní podmínku, tedy zdali algoritmus správně rozeznal, zdali lze řetězec rozdělit na slova.
Pro vstup, který lze rozdělit mezerami existuje posloupnost indexů, které vstupní řetězec dělí. Náš algoritmus by v podobě bez memoizace prošel všechny možné kombinace slov, které lze z řetězce získat, takže by musel řešení najít. Musíme ještě dokázat, že memoizace nás nepřipraví o žádné informace. Trochu neintuitivní je, zdali nemůžeme daný řetězec rozdělit tak, abychom vzali něco, co jsme zahrnuli do memoizace, ve které je třeba informace, že tento podřetězec rozložit nejde. Nicméně k tomuto je v algoritmu přítomen for cyklus, který zkusí všechny prefixy, přičemž uložené informace využíváme až na sufixy. Tudíž stále vyzkoušíme všechny možné kombinace rozdělení a k řešení se dostaneme.

Jméno: Karel Kubíček UČO: 408351

Jméno: Henrich Lauko UČO: 410438

K výstupu False se můžeme dostat jedině v případě, že všechny možné rozdělení projdeme. V pomocném poli můžou i v takovém případě vyjít na některých pozicích hodnoty True, nicméně k těmto hodnotám podslovům neexistuje takový prefix, který by se ze slov skládal.

Asymptotická časová složitost algoritmu patří do třídy $\mathcal{O}(n^2)$ a algoritmus má navíc lineární paměťové nároky. Zatímco paměťová složitost je jasná, důvod proč je časová složitost polynomiální nemusí být zcela jasný. Je zřejmé, že for cyklus provede $\mathcal{O}(n)$ průběhů, kde nekonstantní složitost má jen rekurzivní volání. Díky memoizaci a postupu od konce se však rekurzivní volání může volat maximálně do hloubky 1. To je zajištěno tím, že rekurze se volá nejdříve u podřetězce délky 2, a až potom u podřetězce délky 3. Obecně tedy platí, že v případě volání podřetězce délky $i+1$ už máme všechny podřetězce délky maximálně i vyřešené a uložené v poli. Rekurze se tedy zavolá maximálně n-krát, což lze jednoduše dokázat tím, že každým voláním rekurze vyplníme jeden prvek pomocného pole, které má délku n. Lineární počet rekurzivních zanoření a počet průběhů ve for cyklu odpovídající délce řetězce nám dává Gaussův vzorec pro sumu čísel od 1 do n, kde je převládající kvadratický člen.