



## Referát č.9: Dynamické programovanie

## 1. Zadanie

Navrhните a detailne popíšte polynomiálny algoritmus, ktorý rieši nasledujúci problém:

**VSTUP:** Bezkontextová gramatika  $G$  v Chomského normálnej forme a slovo  $w$ .

**VÝSTUP:** Celkový počet rôznych derivačných stromov, ktoré odpovedajú deriváciám slova  $w$  v gramatike  $G$ .

## 2. Dynamické programovanie

Dynamické programovanie slúži hlavne na optimalizáciu rekurzívneho riešenia problémov. Jeho myšlienkou je uložiť výsledky čiastkových problémov, aby v prípade potreby nebolo nutné ich opäť počítať. Práve táto optimalizácia zlepšuje časovú zložitosť z exponenciálnej na **polynomiálnu**. V prípade riešenia problému Fibonacciho postupnosti sa jedná dokonca o zlepšenia na lineárnu.

## 3. Chomského normálna forma

Ako prvé by som vysvetlila to, ako vôbec vyzerá gramatika v Chomského normálnej forme.

Gramatika je v Chomského normálnej forme, ak každé jej pravidlo je tvaru  $X \rightarrow YZ$  alebo  $X \rightarrow a$ , kde  $X, Y, Z$  označujú neterminály a  $a$  je terminálny symbol. Teda, na ľavej strane je jeden neterminál a na pravej buď dva neterminály alebo terminál. Posledné pravidlo hovorí, že epsilon môže byť len na pravej strane **počiatočného** neterminálu a ak tam je, žiadny iný neterminál sa už nemôže na počiatočný prepísať.

### 3.1. Príklad gramatiky

$S \rightarrow AB \mid BC$

$A \rightarrow BA \mid a$

$B \rightarrow CC \mid b$

$C \rightarrow AB \mid a$

## 4. Cocke-Younger-Kasami algoritmus

Rieši problém toho, či slovo patrí do jazyka zadaného práve Chomského normálnou formou pomocou dynamického programovania. Je založený na princípe, že riešenie úlohy  $[i, j]$  je možné zostaviť z riešenia podproblému  $[i, k]$  a podproblému  $[k, j]$ .

### Popis fungovania algoritmu:

Pre reťazec dĺžky  $N$  zostrojí tabuľku s veľkosťou  $N \times N$ . Každá bunka  $[i, j]$  v tabuľke je množina všetkých častí, ktoré môžu vytvoriť podreťazec siahajúci od pozície  $i$  po pozíciu  $j$ . Proces zahŕňa vyplnenie tabuľky riešeniami čiastkových problémov, ktoré sa vyskytujú v parsovaní zdola-nahor. Bunky sú teda vyplnené zľava doprava a zdola nahor.

Konkrétne môžeme povedať, že v prvom kroku sa všetky terminály prepíšu na odpovedajúce neterminály. Následne sa prechádzajú podreťazce od dĺžky 2 až po dĺžku celého slova. Keďže sa využíva dynamické programovanie a ukladanie čiastkových výsledkov do tabuľky, ďalej sa už stačí len pozrieť, či je možné danú časť podreťazca vygenerovať z niektorého neterminálu gramatiky. Napr. podreťazce dĺžky 3 je možné vytvoriť z dvoch podreťazcov a to buď v kombinácii 1, 2 alebo 2, 1 (čísla predstavujú dĺžku). Tieto čísla nám umožnia správne pristúpiť na pozície v tabuľke, z ktorých sa zisťuje, či je možné z daného neterminálu vygenerovať podreťazec. Takto sa postupuje až do dĺžky celého slova. To, či slovo patrí do jazyka sa určí na základe toho, či je v množine neterminálov, ktoré ho generujú počiatočný neterminál.

<b>5 písmen</b>	C, S, A	-	-	-	-
<b>4 písmená</b>	-	A, S, C	-	-	-
<b>3 písmená</b>	-	B	B	-	-
<b>2 písmená</b>	A, S	B	S, C	A, S	-
<b>1 písmeno</b>	B	A, C	A, C	B	A, C
	<b>b</b>	<b>a</b>	<b>a</b>	<b>b</b>	<b>a</b>

## 5. Počítanie derivačných stromov

Na zistenie počtu derivačných stromov daného slova stačí pomerne jednoduchá úprava samotného algoritmu. V momente, keď je vyplňaná tabuľka, si k jednotlivým neterminálom uložíme počet možností, z koľkých mohol daný neterminál vzniknúť. Inak povedané, koľko pravidiel na danom mieste vygenerovalo daný neterminál. Počty derivačných stromov sa rátajú nasledovne:

- v rámci jedného pravidla sa medzi sebou násobia,

- inak sčítavame (keď daný neterminál vygeneruje druhé, resp. x-té pravidlo, pričítame ho k aktuálnej hodnote)

Ak slovo patrí do jazyka, tak pri počiatočnom netermináli bude uložený počet derivačných stromov.

## 6. Zdroje

<http://www.cs.vsb.cz/kot/down/ti2007/TI-text-20070410.pdf>

<https://www.xarg.org/tools/cyk-algorithm/>

[https://www.youtube.com/watch?v=MrC\\_H6p9k8o](https://www.youtube.com/watch?v=MrC_H6p9k8o)