

základ F: Quick sort: Opakování algoritmu, složitost až

QUICKSORT(F, l):

if (F < l):

i = DIVIDE(A, f, l)
QUICKSORT(f, i-1)
QUICKSORT(i+1, l)

DIVIDE(A, f, l):

x = A[l]

i = f - 1

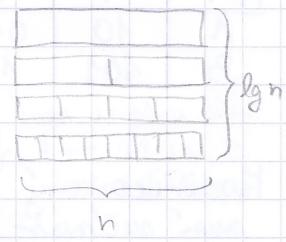
for j in range (f, l):

if A[j] ≤ x

i += 1

swap(A[i], A[j])

return i



$$\text{Výpočet: } T(n) = c \cdot n + \frac{1}{n} \cdot \sum_{i=1}^n (T(i-1) + T(n-i)) = c \cdot n + \frac{1}{n} \left(\sum_{i=1}^n T(i-1) + \sum_{i=1}^n T(n-i) \right) = \\ = cn + \frac{1}{n} \cdot \left(\sum_{j=0}^{n-1} T(j) + \sum_{j=0}^{n-1} T(j) \right) = cn + \frac{2}{n} \cdot \sum_{j=0}^{n-1} T(j)$$

$$T(0) = T(1) = 0$$

Tvrzení: pro $n \geq 2$ platí $T(n) \leq 2c \cdot n \cdot \ln(n)$

Dk. základu indukce:

$$1) n=2, \text{ pak } T(2) = 2c + \frac{2}{2} \cdot \sum_{j=0}^1 T(j) = 2c + T(0) + T(1) = 2c + 0 = 2c = k$$

$$2c \cdot 2 \cdot \ln(2) = 4c \cdot \ln(2) \quad 4c \cdot \ln(2) \geq 2c, \text{ ano, platí'}$$

2) předpokládám, že tvrzení platí pro $\forall m, 2 \leq m \leq n$

$$T(n) = cn + \frac{2}{n} \cdot \sum_{j=0}^{n-1} T(j) = cn + \frac{2}{n} (T(0) + T(1) + \sum_{j=2}^{n-1} T(j)) \leq cn + \frac{2}{n} (T(0) + T(1)) + \\ + \frac{4c}{n} \cdot \sum_{j=2}^{n-1} j \cdot \ln(j)$$

$$\sum_{j=2}^{n-1} j \cdot \ln(j) \leq \int_2^n x \cdot \ln(x) dx = \begin{cases} u = \ln(x) & u' = x \\ w = \frac{1}{x} & w = \frac{x^2}{2} \end{cases} = \int_2^n u \cdot w du = \int_2^n \frac{x^2}{2} \cdot \ln(x) dx = \\ = \left[\frac{x^2}{2} \cdot \ln(x) \right]_2^n - \int_2^n \frac{1}{x} \cdot \frac{x^2}{2} dx = \left[\frac{x^2}{2} \cdot \ln(x) - \frac{x^2}{4} \right]_2^n = \frac{n^2}{2} \ln(n) - \frac{n^2}{4} - (2 \ln(2) - 1) \leq \\ \leq \frac{n^2}{2} \ln(n) - \frac{n^2}{4}$$

$$\text{Proto v indukci použijeme } cn + \frac{4c}{n} \sum_{j=2}^{n-1} j \cdot \ln(n) \leq cn + \frac{4c}{n} \left(\frac{n^2}{2} \ln(n) - \frac{n^2}{4} \right) = \\ = cn + 2c \cdot n \cdot \ln(n) - cn = 2c \cdot n \cdot \lg(n)$$

Při vhodné volbě pivotu je třeba $n \cdot \lg n$ kroků, při správné volbě n^2 kroků. Asymptoticky to ale vychází $n \cdot \lg(n)$ rozdíly.

PCP

Postov korespondenční problém: jsou daný dva řešení sítí A, B nad danou alfabetem Σ_i . $A = (w_1, w_2, \dots, w_r), B = (x_1, x_2, \dots, x_r)$, kde $w_i, x_i \in \Sigma_i^*$, $i = 1, 2, \dots, r$. Rekognize, že dvojice A, B má řešení, jestliže existuje posloupnost i_1, i_2, \dots, i_r indexů, když $j \in \{1, 2, \dots, r\}$ lze říct:

$$w_{i_1}, w_{i_2}, \dots, w_{i_r} = x_{i_1}, x_{i_2}, \dots, x_{i_r}$$

Přednáška
21.5.2019

Obráka: Existuje řešení dané instance?

F:	1	2	3
A	1	10111	10
B	111	10	0

Ano, řešení existuje:

A:	1	0	1	1	1	1	1	0
B:	1	0	1	1	1	1	1	0

2 1 1 3
je řešení.

Př:	1	2	3	1	3	3	3...
A	10	011	101	A	10101101		
B	101	11	011	B	101011011		

Taže instance nemá řešení.

nemůžeme skončit

Modifikovaný PCP (MPCP): Téměř stejný jako PCP, ale je dáno, že řešení musí začínat prvním slovem. $\exists j \quad i_1 = 1$.

Věta: Platí $UN \leq MPCP \leq PCP$.

$MPCP \leq PCP$ na příkladu

	1	2	3
A	10111	1	10
B	10	111	0

Tažebka je minimální slovy, ale změnili jsme sloupce 2 a 1, aby to bylo MPCP.

nová tažebka

	0	1	2	3	4
C	*y ₁	y ₁	y ₂	y ₃	#
D	z ₁	z ₁	z ₂	z ₃	*#

Tím, že všechna y_i začínají na znaku z Σ a z_i začínají *, může řešení začínat pouze prvním sloupcem, jinde se totiž * nebudou shodovat.

Slopec 1 musí mít vždy začínat i bez *, protože kdyby tam nebyl, nemohli bychom najít řešení sloh, které ho poslouží některému.

Slopec 4 jsme přidali, alyčkou mohli na konci navrátit slovo z D, které nemá na konci hvězdičky.

Redukce je polynomická, ale to je méně také jeho; protože má rájová rozhodnutelnost.

Řešení sestrojního PCP

0	2	2	3	4
C	*1 *0 *1 *1 *1 *	1 *	1 *	1 *0 *
D	*1 *0 *1 *1 *1	*1 *1 *1 *1	*0	*

$UN \leq MPCP$

$\langle M \rangle w \rightsquigarrow A, B$

$w \in L(M)$ iff (A, B) má řešení

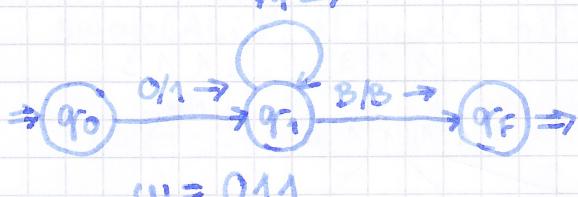
K dané instanci, když TM $M = (Q, \Sigma, \Gamma, S, q_0, B, F)$ a slovo w sestavíme dvojici A, B, která bude mít řešení, právě když M půjčíma w. Do A, B dáme následující pravidla:

A	#	X	#	q X	Z q X	q #	Z q #	X q Y	X q Y q q ##
B	#q ₀ w #	X	#	Y _p	p Z Y	Y _p #	p Z Y #	q	q q #
	$\forall X \in \Gamma^*$				$\delta(q, X) = (p, Y, R)$	$\delta(q, X) = (p, Y, L)$	posun doprava, posun dolera,		
								hlava čte B	$\forall p \in F, X, Y \in \Gamma$

\Leftrightarrow posun doprava

$\forall Z \in \Gamma$

Př: Běžíme stroj, který půjčíma slova $01^n \in \Sigma = \{0, 1\}$, a příprava je na 1^{n+1} .
 $q_1 \rightarrow$

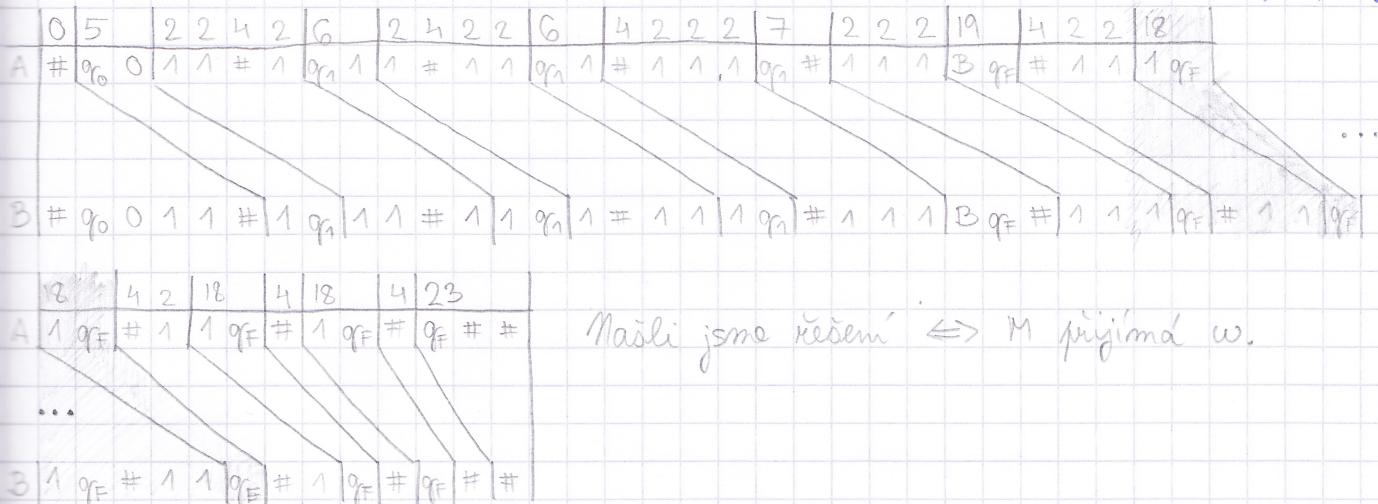


0	1	2	3	4	5	6	7
A	#		0 1 B # q_0 q_1 1 q_F #				
B	#q_0 0 1 1 #	0 1 B # 1 q_1 1 q_F #					

8	9	10	11	12	13
A	0 q_F 0 0 q_F 1 0 q_F B 1 q_F 0 1 q_F 1 1 q_F B				
B	q_F q_F q_F q_F q_F q_F q_F				

M	15	16	17	18	19	20	21	22	23
A	B q _F 0	B q _F 1	B q _F B	0 q _F	1 q _F	B q _F q _F 0	q _F 1	q _F B q _F #	#
B	q _F	q _F	q _F	q _F	q _F	q _F	q _F	q _F	#

Réšení: Vlastně simulujeme práci M nad w: $q_0 011 \rightarrow 1q_1 11 \rightarrow 11q_2 1 \rightarrow 111q_3 B \rightarrow 111Bq_F$



Nášli jsme řešení $\Leftrightarrow M$ přijímá w.

Připomnětí gramatik - už si je možné napamatovat a v dalších se má mít plají.

Gramatika G je čtverice (N, Σ, P, S) , kde N je množina neterminálů, Σ je konečná množina terminálů, S je startovací symbol a P je množina pravidel.

Chomského hierarchie:

Jazyky typu 0 (L_0): Jazyk je typu L_0 , když existuje gramatika typu 0, která ho generuje. Gramatika je typu 0, když má pravidla ve tvaru $(\Sigma \cup N)^* N (\Sigma \cup N)^*$ $\xrightarrow{*} (\Sigma \cup N)^*$.

L_1 : gramatika typu 1 má pravidla ve tvaru $\alpha A \beta \rightarrow \alpha \gamma \beta$, kde $\alpha, \beta \in (N \cup \Sigma)^*$, $\gamma \in (N \cup \Sigma)^+$, $A \in N$. L_1 jsou kontextové jazyky.

L_2 : gramatika typu 2 má pravidla ve tvaru $A \rightarrow B$, kde $A \in N$, $B \in (N \cup \Sigma)^*$. Naučí se bezkontextové jazyky něco CF jazyků.

L_3 : gramatika typu 3 má pravidla ve tvaru $X \rightarrow wY, X \rightarrow w$, kde $X, Y \in N$, $w \in \Sigma^*$. Naučí se regulární jazyky.

O slově můžeme říct, že ho gramatika G generuje nebo negeneruje. Pokud existuje 2 různé derivativní stromy (spolu s odvození slova), nazkame, že G je vícenáčná. Rovnoukdy, když je gramatika vícenáčná, je tento problém:

PCP < vícenáčnost gramatiky.

Definice: Když A, B sestavíme gramatiku G , bude vícenáčná, iFF A, B má řešení.

Ze sestavy A vyvojíme gramatiku $G_A = (f(S_A), \Sigma, \cup \{a_1, a_2, \dots, a_k\}, S_A, P_A)$

P_A : $S_A \rightarrow w_1 S_A a_1 | w_2 S_A a_2 | \dots | w_k S_A a_k$] určitě mají slovo $w_1 w_2 \dots w_k a_1 a_2 \dots a_k$
 $S_A \rightarrow w_1 a_1 | w_2 a_2 | \dots | w_k a_k$

Obdobně vyvojíme $G_B = (f(S_B), \Sigma, \cup \{a_1, a_2, \dots, a_k\}, S_B, P_B)$

P_B : $S_B \rightarrow x_1 S_B a_1 | x_2 S_B a_2 | \dots | x_k S_B a_k$

$S_B \rightarrow x_1 a_1 | x_2 a_2 | \dots | x_k a_k$

Jak a G_B spojíme: $G = (f(S, S_A, S_B), \Sigma, \cup \{a_1, a_2, \dots, a_k\}, S, P)$, $P = P_A \cup P_B \cup \{S \rightarrow S_A, S \rightarrow S_B\}$

Gramatika G je vícenáčná, když $w \dots w a \dots a = x \dots x a \dots a$. Ráky a; mimo, že jsme uživali slova se stejnými místy v sestavách, a checene, aby výsledky byly taky shodné.

Vypočetí jsme ukázali, že mají slovo v prvních dvou gramatik je algoritmicky triviální.