



アルゴリズムとデータ構造III

3回目: 10月16日

構文解析 CYK法

授業資料 <http://ir.cs.yamanashi.ac.jp/~ysuzuki/algorithm3/index.html>

授業の予定(中間試験まで)

1	10/02	スタック(後置記法で書かれた式の計算)
2	10/09	チューリング機械, 文脈自由文法
3	10/16	構文解析 CYK法
4	10/23	構文解析 CYK法
5	10/30	構文解析 CYK法
6	11/06	構文解析 チャート法
7	11/13	グラフ(動的計画法, ダイクストラ法, DPマッチング)
8	11/20	グラフ(DPマッチング, ビームサーチ, A*アルゴリズム)
9	11/27	中間試験

授業の予定(中間試験以降)

10	12/04	全文検索アルゴリズム (simple search, KMP)
11	12/11	全文検索アルゴリズム (BM, Aho-Corasick)
12	12/18	全文検索アルゴリズム (Aho-Corasick), データ圧縮
13	01/08	暗号 (黄金虫, 踊る人形) 符号化 (モールス信号, Zipfの法則, ハフマン符号) テキスト圧縮
14	01/15	テキスト圧縮 (zip), 音声圧縮 (ADPCM, MP3, CELP), 画像圧縮 (JPEG)
15	01/29	期末試験



本日のメニュー

- スタック(復習)
 - Z80シミュレータの動作
 - ソフトウェアの紹介
 - メモリ領域
- 構文解析
 - 構文木(急いで走る一郎を見た)
- CYK法
 - CYKアルゴリズムの説明
 - 解析例(急いで走る一郎を見た)

7 2 3 + - を計算してみよう

(アセンブリ言語でプログラミング)

数式(7 2 3 + -)をメモリ(データ領域)に書き込まれている

3. データ領域から1文字読み込む

1. 数字(アスキーコード:30H~39H)なら

- 数値に変換し, 数値をスタックにプッシュ

2. 演算子なら

1. 一旦スタックにプッシュし, ポップする.

2. スタックからポップし, 数値をBレジスタに取り込む

3. スタックからポップし, 数値をAレジスタ(アキュムレータ)に取り込む

4. 演算子が+なら

- A + B を計算し, Aレジスタに計算結果を格納

5. 演算子が-なら

- A - B を計算し, Aレジスタに計算結果を格納

6. Aレジスタの内容をスタックにプッシュ

4. データ領域すべてを読み終えるまで続ける.

簡単な計算の例 7 2 3 + -

```
;後置記法 7 2 3 + - の計算
ORG 8000H ;
LD HL, DATA ; 数式の先頭番地を指定
LOOP: LD A, (HL)
      CP 00H
      JP Z, OWARI ; 数式を全部読み込んだら終わり

      LD E, (HL)
      LD D, 0H
      LD A, (HL)
      INC HL
      CP 2BH
      JP Z, LOOPA ; +なら加算処理へ
      CP 2DH
      JP Z, LOOPS ; -なら減算処理へ
      LD A, E
      SUB 30H ; 数字なら数値に変換
; Aレジスタの内容をスタックへプッシュ
STPUSH: LD E, A
        LD D, 0H
        PUSH DE ; 読み込んだ数値をスタックへプッシュ

        JP LOOP ; つぎの文字読み込みへ
```

```
;加算
LOOPA: PUSH DE ; 演算子をスタックへプッシュ
       POP DE ; 演算子をスタックからポップ
       POP DE ; 数値をスタックからポップ
       LD B, E ; スタックトップの値をBレジスタへ
       POP DE ; 数値をスタックからポップ
       LD A, E ; スタックトップの値をAレジスタへ
       ADD A, B ; 加算( A <= A + B )
       JP STPUSH

; 減算
LOOPS: PUSH DE ; 演算子をスタックへプッシュ
       POP DE ; 演算子をスタックからポップ
       POP DE ; 数値をスタックからポップ
       LD B, E ; スタックトップの値をBレジスタへ
       POP DE ; 数値をスタックからポップ
       LD A, E ; スタックトップの値をAレジスタへ
       SUB B ; 減算( A <= A - B )
       JP STPUSH

;
OWARI: HALT
;入力データ
DATA:  DEFB 37H ;7
       DEFB 32H ;2
       DEFB 33H ;3
       DEFB 2BH ;+
       DEFB 2DH ;-
       DEFB 00H ;END

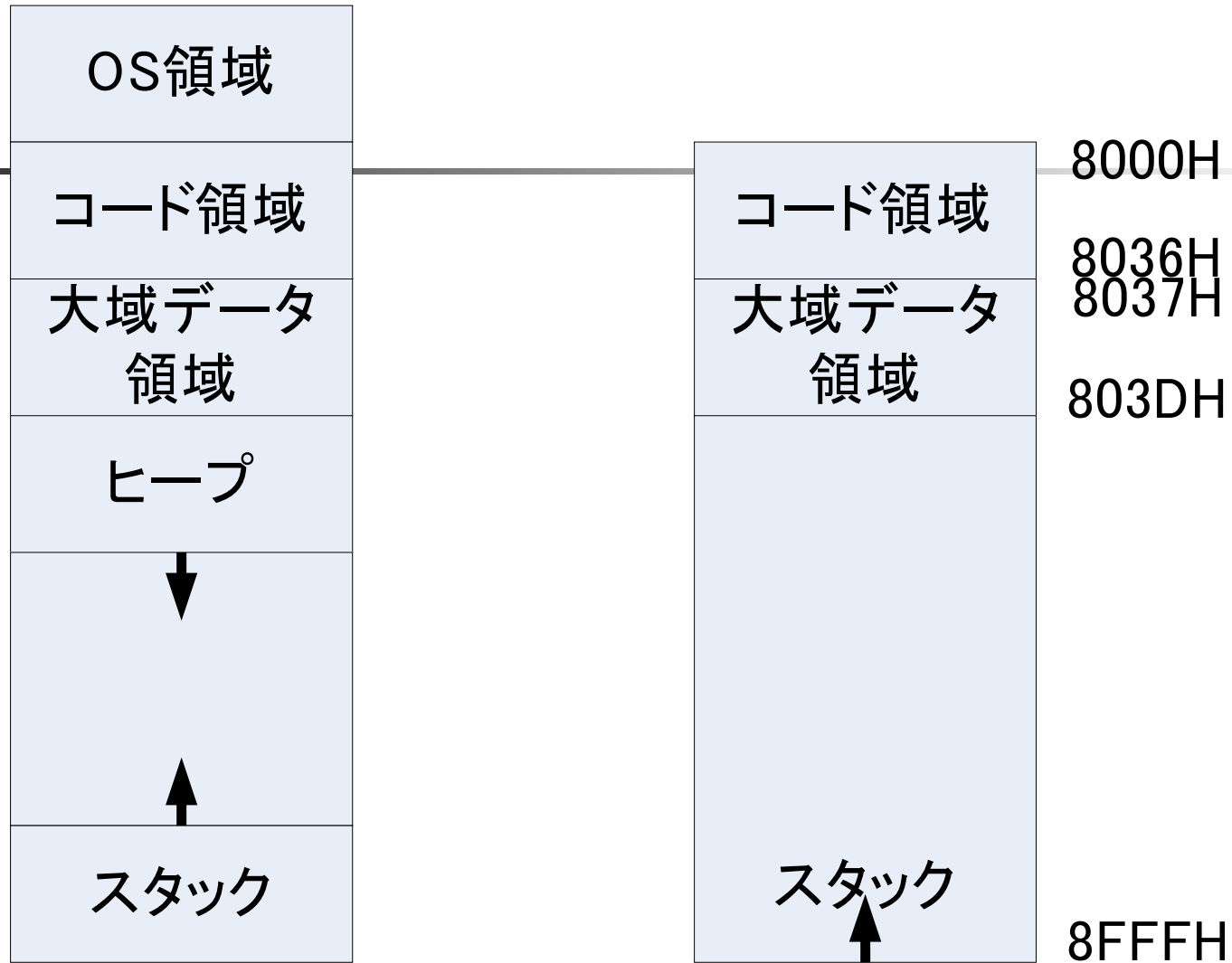
END
```



Z80 シミュレータ

- Z80シミュレータ for Win32
 - <http://www.game3rd.com/soft/z80edit/index.htm>

(スタックを含めた)メモリの様子



メモリ領域の配置例

Z80シミュレータ



構文解析 CYK法

- 文脈自由文法で生成された文から自動的に構文木を生成する.



構文解析とは(Wikipediaより)

- ある文章の文法的な関係を説明すること(*parse*)。計算機科学の世界では、構文解析は字句解析 (*Lexical Analysis*) とともに、おもにプログラミング言語などの形式言語の解析に使用される。また、自然言語処理に応用されることもある。
- コンパイラにおいて構文解析を行う機構を構文解析器 (Parser) と呼ぶ。
- 構文解析は入力テキストを通常、木構造のデータ構造に変換し、その後の処理に適した形にする。字句解析によって入力文字列から字句を取り出し、それらを構文解析器の入力として、構文木や抽象構文木のようなデータ構造を生成する。

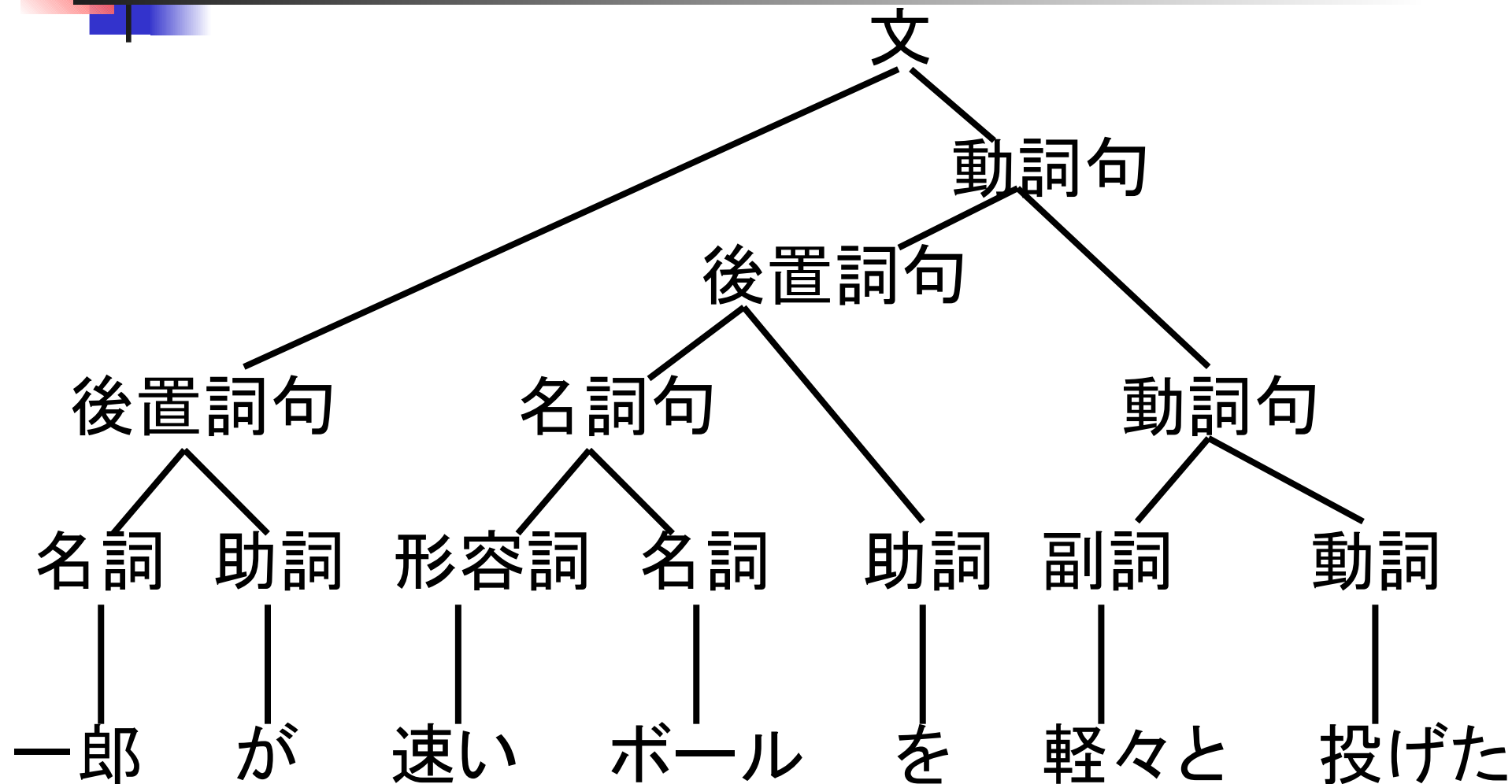


構文解析

- 入力文(記号列)が与えられたとき, 文法によってその文を解析し, その構造を明らかにする
- 代表的な構文解析アルゴリズム
 - **CYK法**
 - チャート法
 - アーリー法
 - LR法

構文木

(一郎が速いボールを軽々と投げた)





CYK (Cocke-Younger-Kasami) 法

- ボトムアップアルゴリズム
- 扱える文法
 - チョムスキーの標準形
 - $A \rightarrow BC$
 - $A \rightarrow a$
- CYK表
 - 構文解析の途中経過を保持するための表



CYKアルゴリズム

- チョムスキーの標準形で表される文脈自由文法を対象とした構文解析法
- チョムスキーの標準形
 - $A \rightarrow BC$ ($A, B, C \in V_n$)
 - $A \rightarrow a$ ($A \in V_n, a \in V_t$)

$X \rightarrow aB$ はチョムスキーの標準形ではないが
 $X \rightarrow AB, A \rightarrow a$ に分解できる

$X \rightarrow ABC$ はチョムスキーの標準形ではないが
 $X \rightarrow AY, Y \rightarrow BC$ に分解できる

チョムスキーの標準形の例 「急いで走る一郎を見る」

A→BC型

A→a型

- (1) $s \rightarrow pp \ v$
- (2) $s \rightarrow adv \ vp$
- (3) $vp \rightarrow pp \ v$
- (4) $vp \rightarrow adv \ v$
- (5) $np \rightarrow vp \ n$
- (6) $np \rightarrow v \ n$
- (7) $pp \rightarrow np \ p$
- (8) $pp \rightarrow n \ p$
- (9) $adv \rightarrow \text{急いで}$
- (10) $n \rightarrow \text{一郎}$
- (11) $p \rightarrow \text{を}$
- (12) $v \rightarrow \text{走る}$
- (13) $v \rightarrow \text{見る}$

CYK構文解析の概要

1. 急いで 2. 走る 3. 一郎 4. を 5. 見た

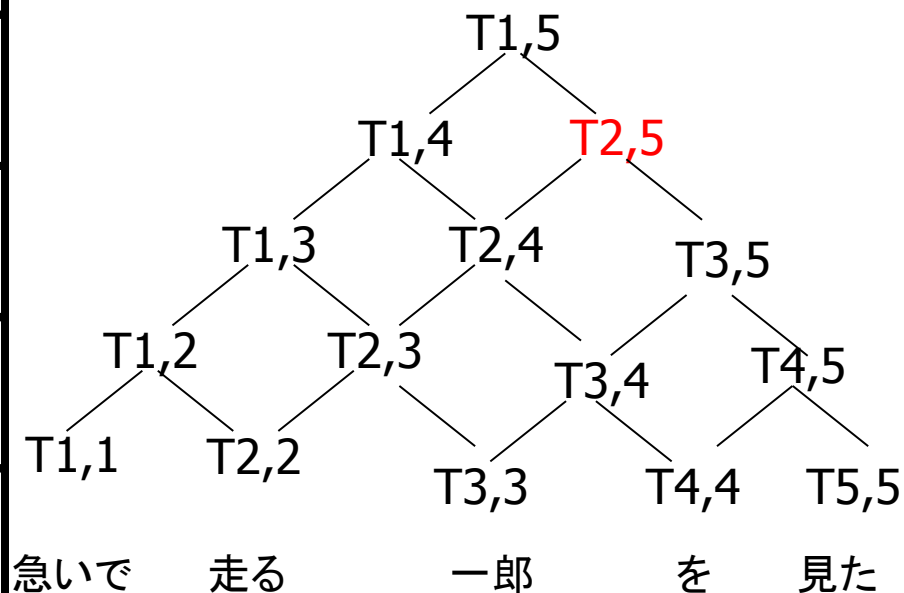
1. 急いで					
2. 走る		T2,2	T2,3	T2,4	T2,5
3. 一郎					T3,5
4. を					T4,5
5. 見た					T5,5

T2,5: 走る一郎を見た

T2,2: 走る | T3,5: 一郎を見た

T2,3: 走る一郎 | T4,5 を見た

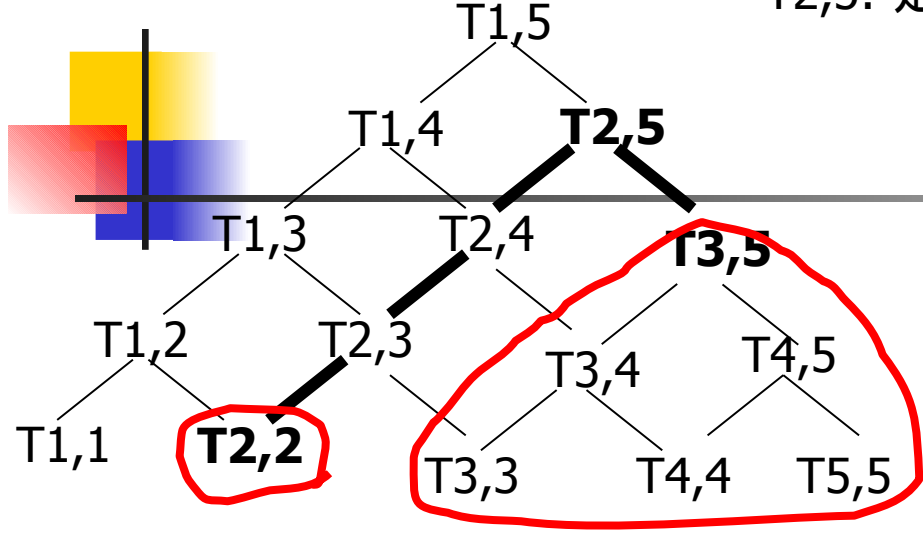
T2,4: 走る一郎を | T5,5 見た



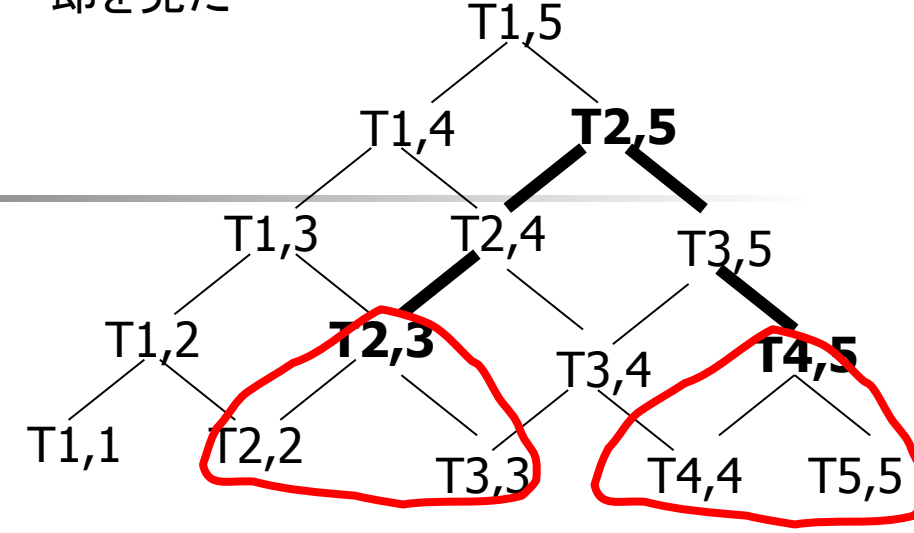
CYK表は構文木を表している

T2,5までの部分木

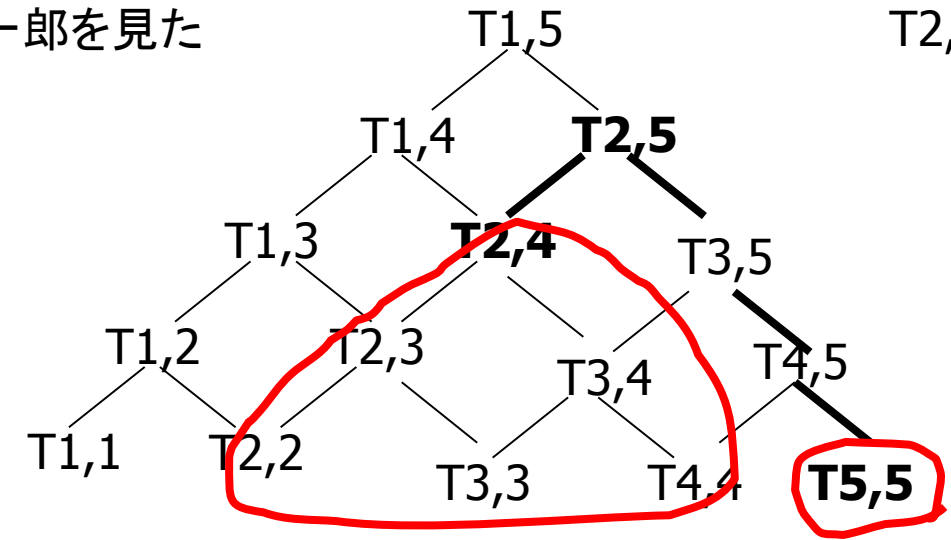
T2,5: 走る一郎を見た



急いで 走る 一郎 を 見た
 T2,2: 走る | T3,5: 一郎を見た



急いで 走る 一郎 を 見た
 T2,3: 走る一郎 | T4,5 を見た



急いで 走る 一郎 を 見た
 T2,4: 走る一郎を | T5,5 見た



CYKアルゴリズム

1. $A \rightarrow a$ の生成規則を用いて, 主対角線上の要素を計算

for $i = 1$ to N

$$T_{i,j} = \{A \mid A \rightarrow w_i\}$$

2. $A \rightarrow BC$ の生成規則を用いて, 2番目以降の対角線上の要素を計算

for $n = 1$ to $N - 1$

for $i = 1$ to $N - n$

$$T_{i,i+n} = \bigcup_{j=1}^n \{A \mid A \rightarrow BC, B \in T_{i,i+j-1}, C \in T_{i+j,i+n}\}$$

3. $S \in T_{1,N}$ であれば, $w_1 \cdots w_N$ は開始記号 S から導出可能

CYK構文解析表(完成)

1. 急いで 2. 走る 3. 一郎 4. を 5. 見た

1. 急いで	adv→急いで	vp→adv v	np→vp n	pp→np p	vp→pp v s→pp v s→adv vp
2. 走る		v→走る	np→v n	pp→np p	vp→pp v s→pp v
3. 一郎	<ul style="list-style-type: none"> ■ (1) s→pp v ■ (2) s→adv vp 		n→一郎	pp→n p	vp→pp v s→pp v
4. を	<ul style="list-style-type: none"> ■ (3) vp→pp v ■ (4) vp→adv v 			p→を	
5. 見た	<ul style="list-style-type: none"> ■ (5) np→vp n ■ (6) np→v n ■ (7) pp→np p ■ (8) pp→n p 	<ul style="list-style-type: none"> ■ (9) adv→急いで ■ (10) n→一郎 ■ (11) p→を ■ (12) v→走る ■ (13) v→見る 			v→見た

CYK構文解析表(1/5)

1. 急いで 2. 走る 3. 一郎 4. を 5. 見た

1. 急いで	adv→急いで				
2. 走る		v→走る			
3. 一郎			n→一郎		
4. を				p→を	
5. 見た					v→見た

■ (1) s→pp v	■ (9) adv→急いで
■ (2) s→adv vp	■ (10) n→一郎
■ (3) vp→pp v	■ (11) p→を
■ (4) vp→adv v	■ (12) v→走る
■ (5) np→vp n	■ (13) v→見る
■ (6) np→v n	
■ (7) pp→np p	
■ (8) pp→n p	

CYK構文解析表(2/5)

1. 急いで 2. 走る 3. 一郎 4. を 5. 見た

1. 急いで	adv→急いで	vp→adv v			
2. 走る		v→走る	np→v n		
3. 一郎	<ul style="list-style-type: none"> ■ (1) s→pp v ■ (2) s→adv vp ■ (3) vp→pp v 		n→一郎	pp→n p	
4. を	<ul style="list-style-type: none"> ■ (4) vp→adv v ■ (5) np→vp n 			p→を	
5. 見た	<ul style="list-style-type: none"> ■ (6) np→v n ■ (7) pp→np p ■ (8) pp→n p 	<ul style="list-style-type: none"> ■ (9) adv→急いで ■ (10) n→一郎 ■ (11) p→を ■ (12) v→走る ■ (13) v→見る 			v→見た

CYK構文解析表(3/5)

1. 急いで 2. 走る 3. 一郎 4. を 5. 見た

1. 急いで	adv→急いで	vp→adv v	np→vp n		
2. 走る		v→走る	np→v n	pp→np p	
3. 一郎	<ul style="list-style-type: none"> ■ (1) s→pp v ■ (2) s→adv vp ■ (3) vp→pp v 		n→一郎	pp→n p	vp→pp v s→pp v
4. を	<ul style="list-style-type: none"> ■ (4) vp→adv v ■ (5) np→vp n 			p→を	
5. 見た	<ul style="list-style-type: none"> ■ (6) np→v n ■ (7) pp→np p ■ (8) pp→n p 	<ul style="list-style-type: none"> ■ (9) adv→急いで ■ (10) n→一郎 ■ (11) p→を ■ (12) v→走る ■ (13) v→見る 			v→見た

CYK構文解析表(4/5)

1. 急いで 2. 走る 3. 一郎 4. を 5. 見た

1. 急いで	adv→急いで	vp→adv v	np→vp n	pp→np p	
2. 走る		v→走る	np→v n	pp→np p	vp→pp v s→pp v
3. 一郎	<ul style="list-style-type: none"> ■ (1) s→pp v ■ (2) s→adv vp ■ (3) vp→pp v 		n→一郎	pp→n p	vp→pp v s→pp v
4. を	<ul style="list-style-type: none"> ■ (4) vp→adv v ■ (5) np→vp n 			p→を	
5. 見た	<ul style="list-style-type: none"> ■ (6) np→v n ■ (7) pp→np p ■ (8) pp→n p 	<ul style="list-style-type: none"> ■ (9) adv→急いで ■ (10) n→一郎 ■ (11) p→を ■ (12) v→走る ■ (13) v→見る 			v→見た

CYK構文解析表(5/5)

1. 急いで 2. 走る 3. 一郎 4. を 5. 見た

1. 急いで	adv→急いで	vp→adv v	np→vp n	pp→np p	vp→pp v s→pp v s→adv vp
2. 走る		v→走る	np→v n	pp→np p	vp→pp v s→pp v
3. 一郎	<ul style="list-style-type: none"> ■ (1) s→pp v ■ (2) s→adv vp 		n→一郎	pp→n p	vp→pp v s→pp v
4. を	<ul style="list-style-type: none"> ■ (3) vp→pp v ■ (4) vp→adv v 			p→を	
5. 見た	<ul style="list-style-type: none"> ■ (5) np→vp n ■ (6) np→v n ■ (7) pp→np p ■ (8) pp→n p 	<ul style="list-style-type: none"> ■ (9) adv→急いで ■ (10) n→一郎 ■ (11) p→を ■ (12) v→走る ■ (13) v→見る 			v→見た

CYK構文解析表(完成！)

1. 急いで 2. 走る 3. 一郎 4. を 5. 見た

1. 急いで	adv→急いで	vp→adv v	np→vp n	pp→np p	vp→pp v s→pp v s→adv vp
2. 走る		v→走る	np→v n	pp→np p	vp→pp v s→pp v
3. 一郎			n→一郎	pp→n p	vp→pp v s→pp v
4. を				p→を	
5. 見た					v→見た

CYK構文解析表 → 構文木 (s→pp v の構文木)

1. 急いで 2. 走る 3. 一郎 4. を 5. 見た

1. 急いで	adv→急いで	vp→adv v	np→vp n	pp→np p	s→pp v
2. 走る		v→走る			
3. 一郎			n→一郎		
4. を				p→を	
5. 見た					v→見た

CYK構文解析表 → 構文木 (s→adv vp の構文木)

1. 急いで 2. 走る 3. 一郎 4. を 5. 見た

1. 急いで	adv→急いで				s→adv vp
2. 走る		v→走る	np→v n	pp→np p	vp→pp v
3. 一郎			n→一郎		
4. を				p→を	
5. 見た					v→見た

文脈自由文法に基づく構文木

