CYK Algorithm Handout

Every Context-free language can be decided in polynomial time, using the CYK (Cook, Younger, and Kuratowski) dynamic programming algorithm.

Notation: If A is a variable of a context-free grammar with terminal alphabet Σ, we let $L(A)$ denote the set of strings over Σ can can be derived from A.

Unless you jump through hoops, a CNF grammar cannot generate the empty string, so we assume that G generates L
$\{\lambda\}$, *i.e.* all non-empty strings of L.

A Chomsky Normal Form grammar is a CF grammar with only two kinds of productions. The left-hand-side of one of these productions is, of course, a variable. The right-hand-side is either a terminal or two variables.

Example. The language L of non-empty even length palindromes over $\{a, b\}$ is generated by the grammar G below.
$S \to aSa$
$S \to bSb$
$S \to aa$
$S \to bb$
In order to use the CYK algorithm, we need a CNF grammar equivalent to G, such as
$S \to AB$
$S \to AC$
$C \to SA$
$S \to BD$
$D \to SB$
$A \to a$
$B \to b$

0.1 Subproblems of CYK.

Let L be a context-free language, and G a CNF (Chomsky normal form) grammar for L with terminal alphabet Σ. An instance of the membership problem for L is a string $w \in \Sigma^*$. and the question is, whether $w \in L$.

Let $n = |w|$. We write $w = a_1 a_2 \ldots a_n$; w has $\binom{n+1}{2}$ substrings. For any $1 \leq \ell \leq i \leq n$ let $w_{i,\ell} = a_i \ldots a_{i+\ell-1}$, the substring of w of length ℓ starting at the i^{th} symbol of w Note that $w_{i,1} = a_i$.

Let m be the number of variables of G. and let A_p be the p^{th} variable. We assume that $A_1 = S$, the start symbol. Let $\mathcal{S}[p, i, \ell]$ be 1 if $w_{i,\ell} \in L(A_p)$, 0 otherwise. There are $m\binom{n+1}{2}$ subproblems, namely to compute the values of $\{\mathcal{S}[p, i, \ell]\}$

We write the dynamic program CYK in pseudocode.

for all $1 \le p \le m$, $1 \le \ell \le i \le n$
  $\mathcal{S}[p, i, \ell] = \text{false}$
for all $1 \le p \le m$, $1 \le i \le n$
  if $(A_p \to a_i)$ $\mathcal{S}[p, i, 1] = \text{true}$
for all $2 \le \ell \le n$
  for all $1 \le i \le n - \ell$
    for all $i + 1 \le j \le n - \ell + 1$
      for $1 \le p \le m$, $1 \le q \le m$, $1 \le r \le m$
        if $(A_p \to A_q A_r$ and $\mathcal{S}[q, i, j - i]$ and $\mathcal{S}[r, j, \ell - j + i])$
          $\mathcal{S}_{[p,i,\ell]} = \text{true};$
return $\mathcal{S}[1, 1, n]$

Walking Through CYK by Hand

Recall that $V = \{A_1, \ldots A_m\}$ is the alphabet of variables of G. We define $\mathcal{V}[i, \ell]$ to be the set of all variables A_p such that $w_{i,\ell} \in L(A_p)$. In terms of our $\mathcal{S}$ notation, $\mathcal{V}[i, \ell]$ is the set of all variables A_p such that $\mathcal{S}[p, i, \ell]$ is true. Hand execution of CYK consists of computing the sets $\{\mathcal{S}[p, i, \ell]\}$ in order of increasing ℓ. In textbook and internet explanations of CYK, each of those sets is shown inside a box which is an entry of a triangular matrix, since $i + \ell \le n + 2$, and this matrix is oriented in the usual row and column manner. However, I have found it intuitive to rotate the matrix 45 degrees, as in Figure 1 below.

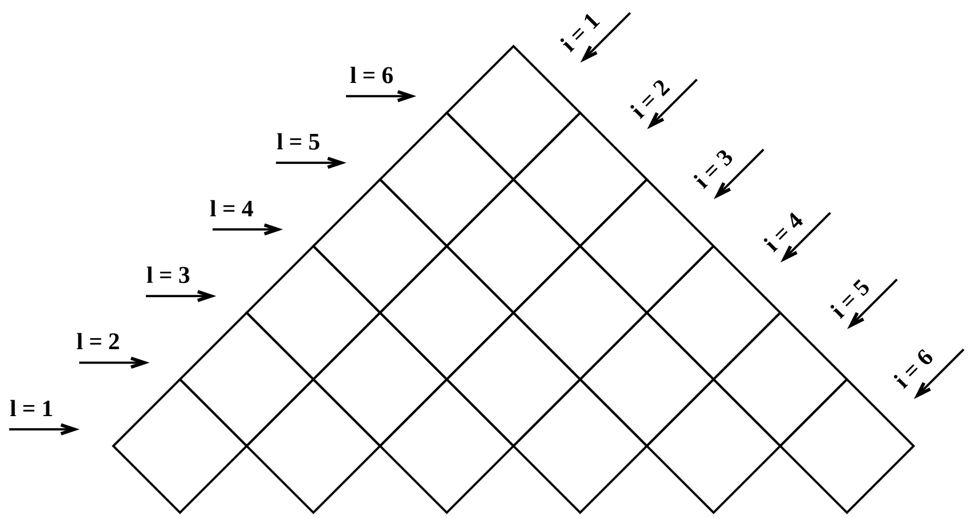

Figure 1: CYK Matrix

Each box corresponds to one substring of w and holds one of the sets $\mathcal{V}[i, \ell]$ The values of those sets are computed from the bottom up: $w \in L$ if and only if S is a member of the top set, $\mathcal{V}[1, n]$

Example. Let G be the CNF grammar:

$S \to IS$

$S \to WS$

$S \to XY$

$X \to IS$

$Y \to ES$

$S \to a$

$E \to e$

$I \to i$

$W \to w$

Here is the CYK matrix with the initial string *iiwaea* written below the first row. The entries of each cell of the matrix are the members of $\mathcal{V}[i, \ell]$ Since S is in the top cell, $w \in L$.

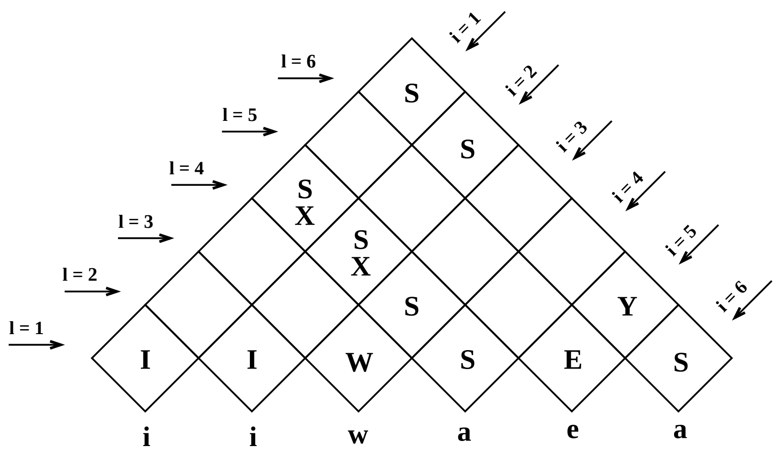

Figure 2: CYK verifying that $iiwaea \in L$.

CYK can then be used to show that the string *ieiaea* is not in L, as shown in Figure 3

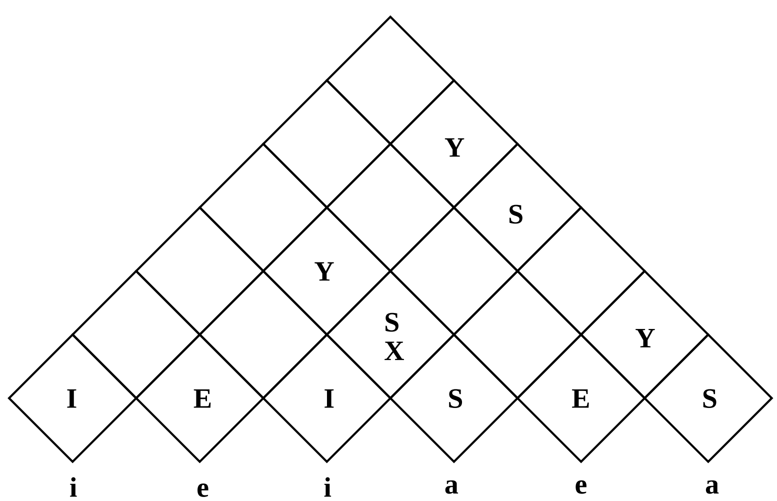

Figure 3: CYK verifying that $iiwaea \notin L$.