Kombinatorische Optimierung

bei Prof. Socher-Ambrosius

Dynamische Programmierung

Algorithmus zur Blocksatzgenerierung

Niels-Peter de Witt Matrikelnr. 2083921
Helge Janicke Matrikelnr. 2083973
Karsten Wolke Matrikelnr. 2083967

2. Juli 2002

Inhaltsverzeichnis

1 Das Problem

Gegeben: Der Algorithmus bekommt folgende Parameter:

- Die Anzahl der Wörter (**Variable n**).
- Die Längen der Wörter $w_1 \dots w_n$ (Wort incl. einem Leerzeichen am Ende). Im weiteren werden die Längen in einem int-Array mit der Bezeichnung **l** gespeichert.
- Die Position, wo der Zeilenumbruch stattfinden soll (**Variable M**). Als Bedingung muß $M \geq \max(l[i])$ mit $i = 0 \dots n$ sein.

Gesucht:

Der Text soll so angeordnet werden, daß der rechte Rand nicht zu sehr flattert.

$r(z) = M - \sum_{k=i}^{j}$ (l[k]) ist der Rest der Zeile z in der die Wörter w_i bis w_j enthalten sind.

Gesucht wird das Minimum folgender Funktion:

$$R = r(z_{line})^2 + r(z_2)^2 + \dots + r(z_{linecount-1})^2$$

z_{line} ist der Zeilenindex und linecount die Anzahl der Zeilen.
Wie man an der Formel erkennen kann, wird die letzte Zeile bei der Betrachtung herausgelassen.

2 Überlegungen

2.1 Wie kann man das Problem umsetzen?

Erste Überlegung:

Das Problem ähnelt ein wenig dem TSP. Wir suchen von einer Position das Minimum zu darauf folgenden Positionen.
Um die gefundenen Minima zu speichern verwenden wir eine Matrix, die als Zeilen die n Wörter beinhaltet und für die Spalten einen Teil der Potenzmenge über die Wörter benutzt.
Einen Teil der Potenzmenge, da es nur einen optimalen Weg von einem Element über dieses Element zu darauf folgenden Elementen geben kann und wir sonst die Wörter in ihrer Anordnung vertauschen würden.

Beispiel: n=4

$m_{i,j}$	$\{1\}$	$\{2\}$	$\{3\}$	$\{4\}$	$\{1,2\}$	$\{2,3\}$	$\{3,4\}$	$\{1,2,3\}$	$\{2,3,4\}$	$\{1,2,3,4\}$
1		X	X	X		X	X		X	
2	X		X	X	X		X	X		X
3	X	X		X	X	X		X	X	X
4	X	X	X		X	X	X	X	X	X

Mit $m_{i,j}$ sind die sind die minimalen Quadratischen Ränder gemeint: $R_{i,j}$

Die Felder, die mit X gekennzeichnet sind, können nie besetzt werden.
Wie man in dieser Tabelle erkennen kann, reicht es, wenn man nur eine Zeile benutzt, da es in jeder Spalte immer nur eine Möglichkeit geben wird um einen Wert einzutragen.
Wir führen nun einen Index **ind** in die Tabelle ein, der später die Mengen ersetzen wird.
Die Tabelle sieht nun folgendermaßen aus:

ind =	0	1	2	3	4	5	6	7	8	9
	$\{1\}$	$\{2\}$	$\{3\}$	$\{4\}$	$\{1,2\}$	$\{2,3\}$	$\{3,4\}$	$\{1,2,3\}$	$\{2,3,4\}$	$\{1,2,3,4\}$
$m_{i,j}$										

2.2 Der Algorithmus zum Blocksatzproblem

Im folgenen suchen wir den Algorithmus, der uns das Minimum sucht.

$$R_{i,j} = \begin{cases} (M - l[i])^2 & \text{falls } i = j \\ \left(M - \sum_{k=i}^{j} l[k]\right)^2 & \text{falls } \left(M - \sum_{k=i}^{j} l[k]\right) \geq 0 \\ \min(\, m_{i,k} + m_{k+1,j} - r(z_{linecount})^2 \,) \quad mit\ i \leq k < j & \text{falls i=1 und j=n} \\ \min(\, m_{i,k} + m_{k+1,j} \,) \quad mit\ i \leq k < j & \text{sonst} \end{cases}$$

Dabei muß folgendes beachtet werden:
In dem Fall, daß i=1 und j=n muß das Minimum aus der Menge der möglichen Kombinationen minus den quadratischen Rest der letzten Zeile $(r(z_{linecount})^2)$ gesucht werden.

Der Algorithmus an einem Beispiel:

Gegeben:

- n = 6
- M = 8
- $l_1 = 5$, $l_2 = 4$, $l_3 = 3$,: $l_4 = 3$, $l_5 = 2$, $l_6 = 7$

Berechnung nach obigem Algorithmus:

In der letzten Spalte ist ein $\Leftarrow$, falls dieser als Minimum erkannt wurde.

ind = 0	$\{1\}$:	$(M-5)^2$	$\Rightarrow$		=	9	
ind = 1	$\{2\}$:	$(M-4)^2$	$\Rightarrow$		=	16	
ind = 2	$\{3\}$:	$(M-3)^2$	$\Rightarrow$		=	25	
ind = 3	$\{4\}$:	$(M-3)^2$	$\Rightarrow$		=	25	
ind = 4	$\{5\}$:	$(M-2)^2$	$\Rightarrow$		=	36	
ind = 5	$\{6\}$:	$(M-7)^2$	$\Rightarrow$		=	7	
ind = 6	$\{1,2\}$:	$\{1\}+\{2\}$	$\Rightarrow$	9+16	=	25	
ind = 7	$\{2,3\}$:	$(M-7)^2$	$\Rightarrow$		=	1	
ind = 8	$\{3,4\}$:	$(M-6)^2$	$\Rightarrow$		=	4	
ind = 9	$\{4,5\}$:	$(M-5)^2$	$\Rightarrow$		=	9	
ind = 10	$\{5,6\}$:	$\{5\}+\{6\}$	$\Rightarrow$	36+7	=	43	
ind = 11	$\{1,2,3\}$:	$\{1\}+\{2,3\}$	$\Rightarrow$	9+1	=	10	$\Leftarrow$
		$\{1,2\}+\{3\}$	$\Rightarrow$	25+25	=	50	
		$\{1\}+\{2,3\}$ ergibt Minimum = 10					
ind = 12	$\{2,3,4\}$:	$\{2\}+\{3,4\}$	$\Rightarrow$	16+4	=	20	$\Leftarrow$
		$\{2,3\}+\{4\}$	$\Rightarrow$	1+25	=	26	
		$\{2\}+\{3,4\}$ ergibt Minimum = 20					
ind = 13	$\{3,4,5\}$:	$(M-8)^2$	$\Rightarrow$		=	0	
ind = 14	$\{4,5,6\}$:	$\{4\}+\{5,6\}$	$\Rightarrow$	25+43	=	68	
		$\{4,5\}+\{6\}$	$\Rightarrow$	9+7	=	16	$\Leftarrow$
		$\{4,5\}+\{6\}$ ergibt Minimum = 16					

ind = 15	$\{1,2,3,4\}$:	$\{1\}+\{2,3,4\}$	$\Rightarrow$	9+20	=	29	$\Leftarrow$
		$\{1,2\}+\{3,4\}$	$\Rightarrow$	25+4	=	29	
		$\{1,2,3\}+\{4\}$	$\Rightarrow$	10+25	=	35	
		$\{1\}+\{2,3,4\}$ ergibt Minimum = 29					
ind = 16	$\{2,3,4,5\}$:	$\{2\}+\{3,4,5\}$	$\Rightarrow$	16+0	=	16	
		$\{2,3\}+\{4,5\}$	$\Rightarrow$	1+9	=	10	$\Leftarrow$
		$\{2,3,4\}+\{5\}$	$\Rightarrow$	20+36	=	56	
		$\{2,3\}+\{4,5\}$ ergibt Minimum = 10					
ind = 17	$\{3,4,5,6\}$:	$\{3\}+\{4,5,6\}$	$\Rightarrow$	25+16	=	41	
		$\{3,4\}+\{5,6\}$	$\Rightarrow$	4+43	=	47	
		$\{3,4,5\}+\{6\}$	$\Rightarrow$	0+7	=	7	$\Leftarrow$
		$\{3,4,5\}+\{6\}$ ergibt Minimum = 7					
ind = 18	$\{1,2,3,4,5\}$:	$\{1\}+\{2,3,4,5\}$	$\Rightarrow$	9+16	=	25	
		$\{1,2\}+\{3,4,5\}$	$\Rightarrow$	25+0	=	25	
		$\{1,2,3\}+\{4,5\}$	$\Rightarrow$	10+9	=	19	$\Leftarrow$
		$\{1,2,3,4\}+\{5\}$	$\Rightarrow$	29+36	=	65	
		$\{1,2,3\}+\{4,5\}$ ergibt Minimum = 19					
ind = 19	$\{2,3,4,5,6\}$:	$\{2\}+\{3,4,5,6\}$	$\Rightarrow$	16+7	=	23	
		$\{2,3\}+\{4,5,6\}$	$\Rightarrow$	1+16	=	17	$\Leftarrow$
		$\{2,3,4\}+\{5,6\}$	$\Rightarrow$	20+43	=	63	
		$\{2,3,4,5\}+\{6\}$	$\Rightarrow$	10+7	=	17	
		$\{2,3\}+\{4,5,6\}$ ergibt Minimum = 17					
ind = 20	$\{1,2,3,4,5,6\}$:	$\{1\}+\{2,3,4,5,6\}$	$\Rightarrow$	9+17	=	26	
		$\{1,2\}+\{3,4,5,6\}$	$\Rightarrow$	25+7	=	32	
		$\{1,2,3\}+\{4,5,6\}$	$\Rightarrow$	10+16	=	26	
		$\{1,2,3,4\}+\{5,6\}$	$\Rightarrow$	29+43	=	72	
		$\{1,2,3,4,5\}+\{6\}$	$\Rightarrow$	19+7	=	26	
		Minimumsuche folgt im weiteren					

Um nun den quadratischen Rest der letzten Zeile zu errechnen, muß bestimmt werden, welche Wörter sich in dieser befinden, und über diese Wörter der quadratische Rest $r(z_{linecount})^2$ gebildet werden.

Im Obiger Tabelle wurde immer angegeben, welches Minimum gewählt wurde, falls eins gefunden wurde.

Optisch kann man nun nachvollziehen, wo ein Zeilenumbruch passiert, denn jedes + zwischen zwei Mengen steht für ein Umbruch.

Nun um also die letzte Zeile zu bestimmen geht man in der Tabelle wieder rückwärts und betrachtet dabei immer nur die Rechte Teilmenge.

ind = 20	$\{1,2,3,4,5,6\}$: $\{1\}+\{2,3,4,5,6\}$ $\Rightarrow$ 9+17 = 26 $\{2,3,4,5,6\}$ $\{2,3\}+\{4,5,6\}$ $\{4,5,6\}$ $\{4,5\}+\{6\}$ Letzte Wort ist w_6: $r(z_{linecount})^2 = (M-l_6)^2 = 7$ Minimum von $\{1\}+\{2,3,4,5,6\}$ ist somit: 26-7 = 19	$\Leftarrow$
ind = 20	$\{1,2,3,4,5,6\}$: $\{1,2\}+\{3,4,5,6\}$ $\Rightarrow$ 25+7 = 32 $\{3,4,5,6\}$ $\{3,4,5\}+\{6\}$ Letzte Wort ist w_6: $r(z_{linecount})^2 = (M-l_6)^2 = 7$ Minimum von $\{1,2\}+\{3,4,5,6\}$ ist somit: 32-7 = 25	
ind = 20	$\{1,2,3,4,5,6\}$: $\{1,2,3\}+\{4,5,6\}$ $\Rightarrow$ 10+16 = 26 $\{4,5,6\}$ $\{4,5\}+\{6\}$ Letzte Wort ist w_6: $r(z_{linecount})^2 = (M-l_6)^2 = 7$ Minimum von $\{1,2,3\}+\{4,5,6\}$ ist somit: 26-7 = 19	
ind = 20	$\{1,2,3,4,5,6\}$: $\{1,2,3,4\}+\{5,6\}$ $\Rightarrow$ 29+43 = 72 $\{5,6\}$ $\{5\}+\{6\}$ Letzte Wort ist w_6: $r(z_{linecount})^2 = (M-l_6)^2 = 7$ Minimum von $\{1,2,3,4\}+\{5,6\}$ ist somit: 72-7 = 65	
ind = 20	$\{1,2,3,4,5,6\}$: $\{1,2,3,4,5\}+\{6\}$ $\Rightarrow$ 19+7 = 26 Letzte Wort ist w_6: $r(z_{linecount})^2 = (M-l_6)^2 = 7$ Minimum von $\{1,2,3,4,5\}+\{6\}$ ist somit: 26-7 = 19	

Über diese Ergebnisse das Minimum suchen ergibt als optimale Lösung den Wert 19.
Dabei können, wie hier auch, mehrere Wege auf diese Lösung führen.

Um alle Umbrüche zu bestimmen muß die linke und rechte Teilmenge auf einzelne Zeilen aufgeschlüsselt werden.

$\{1,2,3,4,5,6\}$: $\Rightarrow$ $\{1\}+\{2,3,4,5,6\}$
$\{1\}+\{2,3,4,5,6\}$ $\Rightarrow$ $\{1\}+\{2,3\}+\{4,5,6\}$
$\{1\}+\{2,3\}+\{4,5,6\}$ $\Rightarrow$ $\{1\}+\{2,3\}+\{4,5\}+\{6\}$
$\{1\}+\{2,3\}+\{4,5\}+\{6\}$

Ansicht der Wörter in einem M=8 breitem Feld:

w_1	X	X	X
w_2, w_3			X
w_4, w_5	X	X	X
w_6			

2.3 Umsetzung für den Rechner

Wenn man den Algorithmus oben betrachtet, so ergeben sich ein paar Fragen, bei der Umsetzung des Algorithmus in eine dem Computer besser bekannte Form.
Da wäre zum einen das Speicherfeld für die Minima. Wie groß wird es gewählt,

welche Mengen werden eingetragen und wie greift man nachher auf das Feld zu, um die die diversen Minima, die immer wieder benötigt werden, zu erhalten? Zum anderen ist es für den Menschen leichter ersichtlich die Umbrüche zurück zu verfolgen, aber die Maschine hat bisher noch noch keine Merker. Es kennt nur die Minima für die Teilprobleme.

2.3.1 Umsetzung des Feldes für die Minima-Speicherung

Die erste Frage ist, wie groß muß das Feld dimensioniert werden?

In der beteiligten Teilmenge P der Potenzmenge über n ergeben sich nur folgende Mengen:

- n Mengen mit nur einem Element.
- (n-1) Mengen mit zwei aufeinander folgenen Elementen.
- (n-2) Mengen mit drei aufeinander folgenen Elementen.
- $\vdots$
- 1 Menge mit n Elementen.

Betrachtet man die nun P, so erhält man an dem obigen Beispiel mit n=6 folgende Größe: 6+5+4+3+2+1
Für n Wörter ergibt sich die Größe folgendermaßen: $\sum_{k=1}^{n} k$ oder auch $\frac{n*(n+1)}{2}$.

P wird nun nach folgendem Schema in das Minima-Feld eingetragen:

- Zuerst in aufsteigender Folge die Mengen mit nur einem Element $\{1\}, \{2\}, \ldots, \{n\}$ bildet die **Ebene n**.
- Dann in aufsteigender Folge die Mengen mit zwei Elementen $\{1, 2\}, \{2, 3\}, \ldots, \{(n-1), n\}$ bildet die **Ebene n-1**.
- $\vdots$
- Die 2 Mengen mit (n-1) Elementen in aufsteigender Folge $\{1, 2, \ldots, (n-1)\}, \{2, 3, \ldots, n\}$ bildet die **Ebene 2**.
- Die Menge mit allen Elementen $\{1, 2, \ldots, n\}$ bildet die **Ebene 1**.

Beispiel für n=4:

$\{1\}$	$\{2\}$	$\{3\}$	$\{4\}$	$\{1,2\}$	$\{2,3\}$	$\{3,4\}$	$\{1,2,3\}$	$\{2,3,4\}$	$\{1,2,3,4\}$
ind=0	1	2	3	4	5	6	7	8	9
Ebene 4				Ebene 3			Ebene 2		Ebene1

Die ersten n Felder können nun ohne Probleme mit dem Algorithmus direkt gefüllt werden.
Kommen wir auf einen Index größer als n beginnt die Umrechnerei um die passenden Indizes zu bekommen, die die Teilmengen für die Minima-Bestimmung bilden.
Dazu benutzen wir den aktuellen Index **i**, die Große des Feldes **ol** und die aktuelle Ebene **layer** um bei der Minima-Suche die passenden Indizes zu generieren.
Es wird ein linker Startwert bestimmt, der dem Index des ersten Elements der aktuellen Teilmenge entspricht:

$lstart = i - (ol - \frac{layer*layer+layer}{2})$

und ein rechter Startwert, der dem Index der restlichen Teilmenge entspricht:

$rstart = i - layer$

Beispiel:

$n = 4$, $i = 8 \Rightarrow \{2, 3, 4\}$
Aktueller Layer zu Index $i = 8$: $layer = 2$
Größe des Minima-Feldes: $ol = 10$

$lstart = 8 - (10 - \frac{2*2+2}{2}) = 8 - 10 + 3 = 1$

$rstart = 8 - 2 = 6$

Diese beiden Indizes entsprechen $\{2\}$ und $\{3, 4\}$.

Es gibt bei bei der Minimum-Suche zu einem Index (n-layer) mögliche Kombinationen (**jmax**) die überprüft werden müssen.

Fortsetzung Beispiel:

Bei $i = 8$, $n = 4$ und $layer = 2$ gibt es $jmax = (4 - 2) = 2$ mögliche Kombinationen:

$\{2\} + \{3, 4\}$ und $\{2, 3\} + \{4\}$
Die Indizierung der ersten Kombination haben wir mit lstart und rstart. Um nun auch die Indizes der zweiten Kombination zu bekommen wandert lstart immer um die Größe der aktuellen Ebene in der sich lstart befindet nach rechts im Feld und rstart um die Größe seiner aktuellen Ebene nach links im Feld.

$lstart = lstart + aktEbene(lstart)$

$rstart = rstart - aktEbene(rstart)$

Fortsetzung Beispiel:

$\{2\} + \{3, 4\} \Rightarrow lstart = 1$ und $rstart = 6$
$\{2, 3\} + \{4\} \Rightarrow lstart = 1 + 4 = 5$ und $rstart = 6 - 3 = 3$

2.3.2 Umsetzung der Merker für Zeilenumbrüche

Um uns zu merken, wo der Umbruch bei einem Minima sich befindet, erweitern wir das Feld der Minima zu einem 2-Dimensionalen Array mit Zeilenanzahl von 3.
Die erste Zeile ist das gefundene Minimum, die 2. Zeile beinhaltet den Index der

gefunden linken Teilmenge und die 3. Zeile den Index der gefundenen rechten Teilmenge.
Die ersten n Elemente bekommen dort in der 2. und 3. Spalte den ihren negativen Indexwert, da sie keine linke und rechte Teilmenge besitzen.
Wenn eine Teilmenge komplett in eine Zeile paßt, so wird in der in der 2. Zeile der negative Index des ersten Elements der Teilmenge und in die 3. Zeile der negative Index des letzten Elements eingetragen.

Beispiel: Tabelle des Beispiels mit n=6:
Die erste Zeile dieser Tabelle entspricht dem Index bzw. der Teilmenge, die 2. Zeile ist das gefundene Minimum, die 3. Zeile der Index der linken Teilmenge und die 4. Zeile der Index der rechten Teilmenge.

0	1	2	3	4	5	6	7	8	9	10
$\{1\}$	$\{2\}$	$\{3\}$	$\{4\}$	$\{5\}$	$\{6\}$	$\{1,2\}$	$\{2,3\}$	$\{3,4\}$	$\{4,5\}$	$\{5,6\}$
9	16	25	25	36	1	25	1	4	9	37
0	-1	-2	-3	-4	-5	0	-1	-2	-3	4
0	-1	-2	-3	-4	-5	1	-2	-3	-4	5

11	12	13	14	15	16	17
$\{1,2,3\}$	$\{2,3,4\}$	$\{3,4,5\}$	$\{4,5,6\}$	$\{1,2,3,4\}$	$\{2,3,4,5\}$	$\{3,4,5,6\}$
10	20	0	10	29	10	1
0	1	-2	9	0	7	13
7	8	-4	5	12	9	5

18	19	20
$\{1,2,3,4,5\}$	$\{2,3,4,5,6\}$	$\{1,2,3,4,5,6\}$
19	11	19
0	7	0
16	14	19

Hier kann man nun anhand der Indizes der linken und rechten Teilmenge rekursiv vom letzten Eintrag zurckgehen und die Zeilenumbrüche bestimmen.

ind = 20:	l = 0	und	r = 19	
ind = 0:	l = 0	und	r = 0	$\Rightarrow$ nach w_1 (ind=0) erfolgt ein Umbruch.
ind = 19:	l = 7	und	r = 14	
ind = 7:	l = -1	und	r = -2	$\Rightarrow$ nach w_2, w_3 (ind=2) erfolgt ein Umbruch.
ind = 14:	l = 9	und	r = 5	
ind = 9	l = -3	und	r = -4	$\Rightarrow$ nach w_4, w_5 (ind=4) erfolgt ein Umbruch.
ind = 5	l= -5	und	r = -5	$\Rightarrow$ nach w_6 (ind=5) erfolgt ein Umbruch.

A Quellcode

```
/**
 *  Description of the Class
 *
 * @author     Helge Janicke, Karsten Wolke, Niels-Peter de Witt
 * @created    4. Mai 2002
 */
public class WC02 {
    /**  Description of the Field */
    public final static int SPACE = 0;
    /**  Description of the Field */
    public final static int BEGIN = 1;
    /**  Description of the Field */
    public final static int END = 2;



    /**
     *  Description of the Method
     *
     * @param  l  Array with the length of each word
     * @param  M  Linebreak. Must be greater or equal the longest word.
     * @return    Description of the Return Value
     */
    public static int opt( int[] l, int M ) {
        int n = l.length;
        int ol = ( n * n + n ) / 2;
        int[][] o = new int[ol][3];
        int layer;
        int cntlayer;
        int tmp;
        int jmax;
        int start;
        int lstart;
        int rstart;
        int length;
        int[] pow = new int[M];

        for ( int i = 0; i < M; i++ ) {
            pow[i] = i * i;
        }

        for ( int i = 0; i < n; i++ ) {
            tmp = M - l[i];
            o[i][SPACE] = pow[tmp];
            o[i][BEGIN] = -i;
            o[i][END] = -i;
        }
```

```
layer = n - 1;
cntlayer = 1;
for ( int i = n; i < ol; i++ ) {
    jmax = n - layer;
    tmp = 0;
    start = i - ( ol - ( layer * layer + layer ) / 2 );

    int jk = 0;

    while ( jk <= jmax && tmp <= M ) {
        tmp += l[start + jk];
        jk++;
    }
    lstart = start;
    rstart = i - layer;
    if ( tmp > M ) {
        o[i][SPACE] = o[lstart][SPACE] + o[rstart][SPACE];
        o[i][BEGIN] = lstart;
        o[i][END] = rstart;

        if ( i == ol - 1 ) {
            // adjust last line
            length = getLengthLastLine( rstart, o );
            o[i][SPACE] -= length;
        }

        for ( int j = 0; j < jmax - 1; j++ ) {
            lstart += n - j;
            rstart -= layer + j + 1;
            tmp = o[lstart][SPACE] + o[rstart][SPACE];

            if ( i == ol - 1 ) {
                // adjust last line
                length = getLengthLastLine( rstart, o );
                tmp -= length;
            }

            if ( tmp < o[i][SPACE] ) {
                o[i][SPACE] = tmp;
                o[i][BEGIN] = lstart;
                o[i][END] = rstart;
            }
        }
    } else {
        tmp = M - tmp;
        //tmp *= tmp;
        o[i][SPACE] = pow[tmp];
        o[i][BEGIN] = -lstart;
        o[i][END] = -( lstart + jmax );
    }
```

```
        if ( layer == cntlayer ) {
            layer--;
            cntlayer = 0;
        }
        cntlayer++;
    }

    for ( int i = 0; i < ol; i++ ) {
        System.err.println( "col=" + i + " l= " + o[i][BEGIN] +
                    " r= " + o[i][END] + " SPACE= " + o[i][SPACE] );
    }

    java.util.Arrays.fill( l, 0 );
    setBr( o[ol - 1][BEGIN], o[ol - 1][END], o, l );
    return o[ol - 1][SPACE];
}


/**
 *  Gets the lengthLastLine attribute of the WC02 class
 *
 * @param  rstart  Description of the Parameter
 * @param  o       Description of the Parameter
 * @return         The lengthLastLine value
 */
private static int getLengthLastLine( int rstart, int[][] o ) {
    if ( o[rstart][END] > 0 ) {
        return getLengthLastLine( o[rstart][END], o );
    }
    return o[rstart][SPACE];
}


/**
 *  Sets the br attribute of the WC02 class
 *
 * @param  l   The new br value
 * @param  r   The new br value
 * @param  o   The new br value
 * @param  br  The new br value
 */
private static void setBr( int l, int r, int[][] o, int[] br ) {
    System.err.println( "l= " + l + " r= " + r );
    if ( r >= 0 && l >= 0 ) {
        if ( o[r][END] > 0 ) {
            System.err.println( "begin=" + l + " end=" + r );
            setBr( o[r][BEGIN], o[r][END], o, br );
        } else {
            br[-o[r][END]] = 1;
        }
```

```
            if ( o[l][END] > 0 ) {
                System.err.println( "lbegin=" + l + " lend=" + r );
                setBr( o[l][BEGIN], o[l][END], o, br );
            } else {
                br[-o[l][END]] = 1;
            }
        }
    }
}
```