Shift Or Algorithmus

- Stringmatching-Algorithmen in C
- Dozent: Max Hadersbeck
- Referentin: Pei Zhao
- Datum: 24.07.2006
- Folien: www.cip.ifi.lmu.de/~zhaop

Inhaltsverzeichnis

Einleitung

- Naives Verfahren: alles, was vor dem Mismatch war, wird vergessen
- Der Knuth-Morris-Pratt-Algorithmus und der Boyer-Moore-Algorithmus sind Optimierungen des Naiven Verfahrens.
- Der Shift-Or-Algorithmus, der auch Baeze-Yates-Gonnet-Algorithmus bzw. Shift-And-Algorithmus genannt wird, geht einen neuen Weg.

Merkmale des Algorithmus

- Nutzt durch Bitoperationen moegliche Parallelisierung
- Theoretischer Hintergrund: NEA
- Praeprozessing: $O(m+\sigma)$
- Searching: $O(n)$
- Erweiterung zu nicht-exaktes Matching

Grundlage

- Eine Menge von Vektoren(Zusta" nden) R_j

$$R_{j+1}[i] = \begin{cases} 1, & \text{falls } i = 0 \\ 1, & \text{falls } R_j[i-1] = 1 \text{ und } Musterzeichen_i = Eingabezeichen_{j+1} \\ 0, & \text{sonst} \end{cases}$$

Bedeutung:

$R_j[i]$ ist genau dann 1, wenn nach Verarbeitung von j zeichen des Textes die letzten i Zeichen mit den i ersten Zeichen des Musters u" berreinstimmen.

5

Grundlage

- Charakteristische Vektoren fu" r alle im Text vorkommende Zeichen,z.B. $S_a[i]$

$$s_a[i] = \begin{cases} 1, & \text{falls im Suchmuster an Stelle } i \text{ das Zeichen } a \text{ steht} \\ 0, & \text{sonst} \end{cases}$$

Beispiel : Suchmuster *abcac*, *Laenge m=5*

Ch.Vektoren:

	s_a	s_b	s_c	s_d	...
a	1	0	0	0	...
b	0	1	0	0	...
c	0	0	1	0	...
a	1	0	0	0	...
c	0	0	1	0	...

Grundlage

- Bitoperation *bitshift* bzw. >> fu" r den Vektor R

$$R = \begin{matrix}0\\0\\0\\0\\0\end{matrix} \rightarrow Bitshift(R) = \begin{matrix}1\\0\\0\\0\\0\end{matrix} \rightarrow Bitshift(R) = \begin{matrix}1\\1\\0\\0\\0\end{matrix} \rightarrow Bitshift(R) = \begin{matrix}1\\1\\1\\0\\0\end{matrix}$$

- Ein Treffer fu" r ein Suchmuster mit Laenge m ist gefunden, falls $R_j[m]=1$.

Der Algorithmus

- Initialisierung: $R_0[0,1,2,\ldots,i] = 0$
- Praeprozessing: Jedem Buchstaben des Textes wird ein Ch. Vektor zugewiesen.
- Searching: $R_j = Bitshift(R_{j-1})\ AND\ S_z$
- Bewertung:

  An der Position=J-m+1 liegt ein Treffer, wobei J die Position ist, wo $R_j[m]$ 1 ist.

Der Algorithmus

- Programmcode in C

Praezessing

```
void preprocess() {
   int i;
   s = (int *) calloc (255, sizeof(int));

   for (i = 0; i < m; ++i) {
         length = 1 << i;
         s[p[i]] |= length;
   }
}
```

Der Algorithmus

- Searching

```
int bitShift (int c) {
  return(c << 1 | 1);
}

void search() {
  int j, r = 0;

  for (j = 0; j < n; ++j) {
        r = bitShift (r) & s[t[j]];

        if (r & length) {
          printf ("Treffer bei Position %d\n", j - m + 2);
        }
  }
}
```

Beispiele

- Muster: *abcac*
- Text: *abcabcac*

$i\backslash$	R_0	$\gg$	s_a	R_1	$\gg$	s_b	R_2	$\gg$	s_c	R_3	$\gg$	s_a	R_4	$\gg$	s_b	R_5	$\gg$	s_c	R_6	$\gg$	s_a	R_7	$\gg$	s_c	R_8
1	0	1	1	1	1	0	0	1	0	0	1	1	1	1	0	0	1	0	0	1	1	1	1	0	0
2	0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	0	0
3	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	0	0	0	0	1	0
4	0	0	1	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	0	0	0
5	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	1	1

Nicht exaktes Matching

- Ein zusaetzliches Zeichen im Text

  z.B. Text: feldsalat Muster: saat

  $R_j^1 = (Bitshift\ (R_{j-1}^1)\ AND\ S_j)\ OR\ R_{j-1}$

- Ein fehlendes Zeichen im Text
- $R_j^k = (Bitshift(R_{j-1}^k)\ AND\ S_j)\ OR\ Bitshift(R_j^{k-1})$
- Ein falsches Zeichen im Text
- $R_j^k = (Bitshift(R_{j-1}^k)\ AND\ S_j)\ OR\ Bitshift(R_{j-1}^{k-1})$

Reference

- [BaeGo89] R. A. Baeza-Yates und G. H. Gonnet: A new approach to text searching,
- Proceedings of the 12th Annual ACM-SIGIR conference on
- Information Retrieval. Cambridge, MA, 1989, S. 168 - 175
- [WuMa91] Sun Wu und Udi Manber: Fast text searching with errors, Technical
- report TR-91-11, University of Arizona, Department of Computer
- Science, 1991. http://www.cs.utk.edu/ cs494/readinglist/
- agrep.1.ps
- http://theo.cs.uni-magdeburg.de/lehre05w/ti2_fern/folien/folien1.pdf
- http://de.wikipedia.org/wiki/Shift-Or-Algorithmus