Tree pattern matching and subset matching in deterministic $\mathcal{O}(nlog^3n)$

Emmanuel Filiot

March 23, 2003

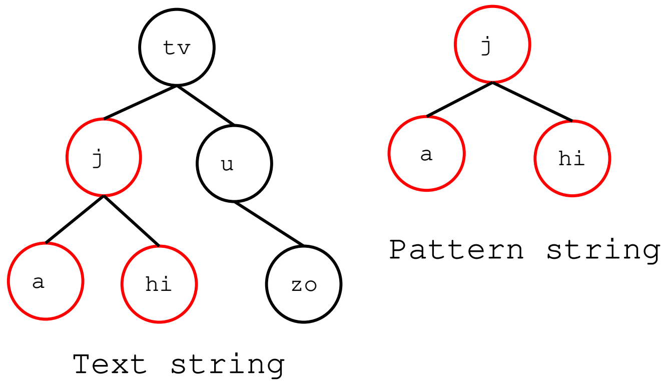
Plan

1.Definitions
2.Overview of the algorithm
3.Superimposed code
4.Minimum weight problem

Definitions (1)

Tree pattern matching problem : Given two ordered trees, called the *text* and the *pattern*, find all occurences of the pattern in the text.
The pattern is said to occurs at a particular position if placing the pattern with root at that text position leads to a situation in which each pattern nod overlaps some text node.

Definitions (2)

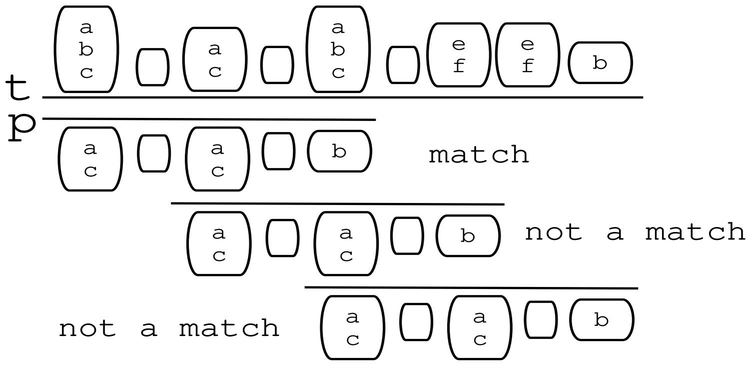


Definitions (3)

Subset matching problem : Given an alphabet $\sum = [0, \ldots, k-1]$, a text string t[0…n-1] and a pattern string p[0…m-1] such that $t[i] \subseteq \sum$, $p[j] \subseteq \sum, \forall i = 0 \ldots n-1, \forall j = 0 \ldots m-1$, output a binary array $\mathcal{T}[0 \ldots n-m]$ such that $\mathcal{T}[i] = 1$ iff $p[j] \subseteq t[i+j] \forall j = 0 \ldots m-1$.

Lemma 1 : Tree pattern matching problem can be reduce to Subset matching problem in $\mathcal{O}(n)$.[CH97]

Subset matching problem



Overview of the algorithm (1)

Notations : $s = \sum_i |t[i]| + \sum_j |p[j]|$ where $|.|$ denotes the cardinality.
Hypothesis : $\bigcup_i t[i] = \bigcup_j p[j]$, $m \leq n \leq 2m$, $s \leq 6m$, i.e. $s = \mathcal{O}(n)$. [a]

[a] As pointed out in [CH97], that can be assumed without loss of generality (otherwise the problem can be linearly reduced to several subset matching subproblems satisfying these properties)

Overview of the algorithm (2)

1. The maximum size of each set of t are minimized by shifting the elements of sets of both t and p in $\mathcal{O}(nlog^3n)$-time → Minimum Weight Problem.
2. The sets of t and p are coded, in $\mathcal{O}(nlog^2n)$-time, into binary vectors of length $\mathcal{O}(log^2n)$ which permit to see if a set is included in another one quickly.
→ Superimposed Codes Problem.
3.The string matching problem is the solved by standard methods based on convolutions in $\mathcal{O}(nlog^3n) - time$.

Superimposed Codes (1)

Data: $S = S_0, ..., S_p \subseteq \mathcal{U}$.
Problem: find equal length binary codes for the elements of each S_i such that $code(S_i)$, defined as the 'or' of the codes of its elements, does not contain any code of elements not in S_i, i.e. :
$\forall e \in \mathcal{U}, \forall S_i \in S$, $e \notin S_i$ implies that there is at least one 1 in $code(e)$ such that the bit in $code(S_i)$ in the same position is a zero.

Superimposed Codes : example

$S_1 = \{a, b, c\}$
$S_2 = \{a, b\}$
$S_3 = \{b, c\}$
$code(a) \; = 1\,1\,0$
$code(b) \; = 0\,0\,0$
$code(c) \; = 0\,1\,1$
$code(S_1) = 1\,1\,1$
$code(S_2) = 1\,1\,0$
$code(S_3) = 0\,1\,1$

Superimposed Codes (3)

It can be noticed that $S_i \subseteq S_j$ iff $code(S_i) \leq code(S_j)$, i.e. $\forall p, code(S_i)_p \leq code(S_j)_p$.

So the problem is to build short codes in a fast time. As the length of the codes is proportional to the set size, the goal of the second problem is to minimize the sizes of the sets.

Minimum Weigth Problem(1)

Motivations : generate string t'' and p'' such that all sets in both t'' and p'' have small cardinality, and p'' matches t'' at position i iff p matches t at the same position.
Notation : $T[a, i] = 1$ where $a = \{a_1, ..., a_q\} \subseteq \mathcal{K}$ and $k \in \{0, 1\}$ iff $T[j, i] = 1$ if $j = a_l$ and 0 otherwise.

As the alphabet $\mathcal{K}$ is finite, it is possible to work with natural integers, instead of characters.

Minimum Weigth Problem(2)

The binary matrix $T[0...k-1, 0...n-1]$ and $P[0...k-1, 0...m-1]$ are defined as follow :

$T[a,i] = 1$ iff $t[i] = a$

$P[a,i] = 1$ iff $p[i] = a$

Condition 1 : p matches t at position i iff :

$\forall j = 0...m-1, P[.,j] \leq T[., i+j]$

Minimum Weigth Problem(3) : Example

$k = 4$, i.e. $\mathcal{K} = \{0, ..., 3\}$

$n = 5,\ m = 3$

$t = \{1,2,3\}, \{1\}, \{0,3\}, \{\}, \{0,2,3\}$

$p = \{2,3\}, \{1\}, \{3\}$

$$T = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \end{pmatrix},\ P = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

p matches t at position 0 .

Minimum Weigth Problem (4)

The weight of a column is the sum of its elements. So the weights of both T and P correspond to the set sizes of both T and P.

Fact : Condition 1 remains invariant when the corresponding rows of both T and P are shifted by the same number of positions.

So it is possible to reduce the number of ones in each column of T, i.e. the set sizes.

Minimum Weight Problem (5)

$$T = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \end{pmatrix},\ P = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

$$T" = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \end{pmatrix},\ P" = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

$t'' = \{2,3\}, \{0,1\}, \{1,3\}, \{0\}, \{2,3\}$ $p'' = \{2,3\}, \{\}, \{1,3\}$

Minimum Weight Problem : Definition

Definition : Given positive integers b,d and a binary matrix $T[0...k-1, 0...n-1]$ containing at most b ones, find (if possible) k positive integers $l_0, \ldots, l_{k-1} \in \{0, \ldots, n-1\}$ such that the matrix T'' obtained by shifting the ith row of T by l_i positions has the property that each of its columns contained at most d ones.

Theorem : Minimum Weight Problem with $b = n$, and $d = \mathcal{O}(n)$ can be solved in deterministic $\mathcal{O}(nlog^3 n)$-time.

Conclusion

- a fast algorithm to solve Tree Pattern Matching Problem
- application in each field which uses tree structure, for example Web information extraction

Bibliography

- R.Cole, R.Hariharan, P.Indyk, "Tree Pattern Matching and Subset Matching in Deterministic $\mathcal{O}(nlog^3 m)$-Time". *In SODA: ACM Symposium on Discrete Algorithms.*
- [CH97] R.Cole, R.Hariharan, "Tree Pattern Matching and Subset Matching in Randomized $\mathcal{O}(nlog^3 m)$ Time", *Proc. STOC'97*, to appear.
- P.Indyk, "Deterministic Superimposed Coding with Application to Pattern Matching", *Proc. FOCS'97.*