Introduction To Algorithm Learning Notes —Chapter 32 (String Matching)

6/23/2009 2:19PM

Automata Terms

The automata theory used a lot of Greek Characters. It it convinient for people in that field, but for newbies, it takes time to get used to that.

Σ means the set of characters.

$\sigma(x) = max\{k : P_k$ is suffix of $x\}$. More exactly, $\sigma(x, P) = max\{k : P_k$ is suffix of $x\}$.

(*Note: P_k is the substring of length k, start from the beginning of string P.)

δ means the state conversion function. $\delta(i, a) = j$ means that the state conversion function δ will convert the state i to state j with the input character a;

$\phi(T_i)$ means the resulted state with the input $T[i]$.

In the automata implementation, consider the example as follows.

$P = ababaca$; $T = abababacaba$.

(*Note: P and T is indexed from 1, not 0 in the text.)

so $\sigma(T)$ is 3, the P_3 is aba.

automata is defined as $\delta(q, a) = \sigma(P_q a)$.

It is similar to the naive implementation, but with some improvement. The idea of the algorithm is as follows: iterate on the input, if everything match the pattern, then we get a match. otherwise, for example, pattern is of size 7, and current imput match the P_5, the sixth position does not match the input a. Now $\delta(q, a)$ is $\delta(5, a)$, suppose it is 2, that means the P_2 is a suffix of current input, we have to start the match from $currentposition - 2 + 1$. because of this, we get the O(n) instead of O(mn).

KMP's method use similar but different way to achieve same result. in KMP's method, if there is any mismatch happened, we know that it is possilbe that the next few shifts are not valid and we do not need to check it. with this knowledge, we can skip a few chars in the input string and improve the performance.

but since the next char still need to be checked, the asymptotic complexity is also $O(n)$ though automata and KMP method have same $O(n)$, but the constand can be different.

Thought: the idea is great and fatastic! it improve the algorithm step by step. it becomes more and more complex, and have better and better performance. It may no longer be the mainstream of Software Engineering, since now most people is focus on application development instead of system research, but it is still interesting and amazing to know.

Notes on the reading of lask weekend. B Tree, B^+ Tree, B^* Tree are all balanced tree, targeted at the indexing on disk storage. the number of braches are high to reduce the height of the tree, it can save disk access effectively.

But I did not get into the details of the B Tree. Not sure how the data structure change effectivly when there are more data added into the tree.