Lexicographically minimal string rotation - Wikipedia, the free encyclopedia

In computer science, the **lexicographically minimal string rotation** or **lexicographically least circular substring** is the problem of finding the rotation of a string possessing the lowest lexicographical order of all such rotations. For example, the lexicographically minimal rotation of "bbaaccaadd" would be "aaccaaddbb". It is possible for a string to have multiple lexicographically minimal rotations, but for most applications this does not matter as the rotations must be equivalent. Finding the lexicographically minimal rotation is useful as a way of normalizing strings. If the strings represent potentially isomorphic structures such as graphs, normalizing in this way allows for simple equality checking.[1] A common implementation trick when dealing with circular strings is to concatenate the string to itself instead of having to perform modular arithmetic on the string indices.

Algorithms[edit]

The Naive Algorithm[edit]

The naive algorithm for finding the lexicographically minimal rotation of a string is to iterate through successive rotations while keeping track of the most lexicographically minimal rotation encountered. If the string is of length n, this algorithm runs in $O(n^2)$ time in the worst case.

Booth's Algorithm[edit]

An efficient algorithm was proposed by Booth (1980).[2] The algorithm uses a modified preprocessing function from the Knuth-Morris-Pratt string search algorithm. The failure function for the string is computed as normal, but the string is rotated during the computation so some indices must be computed more than once as they wrap around. Once all indices of the failure function have been successfully computed without the string rotating again, the minimal lexicographical rotation is known to be found and its starting index is returned. The correctness of the algorithm is somewhat difficult to understand, but it is easy to implement.

```
def lcs(S):
    S += S      # Concatenate string to it self to avoid modular
arithmetic
    f = [-1] * len(S)     # Failure function
    k = 0       # Least rotation of string found so far
    for j in xrange(1,len(S)):
        sj = S[j]
        i = f[j-k-1]
        while i != -1 and sj != S[k+i+1]:
            if sj < S[k+i+1]:
```

```
                k = j-i-1
            i = f[i]
        if sj != S[k+i+1]: # if sj != S[k+i+1], then i == -1
            if sj < S[k]: # k+i+1 = k
                k = j
            f[j-k] = -1
        else:
            f[j-k] = i+1
    return k
```

Of interest is that removing all lines of code which modify the value of k results in the original Knuth-Morris-Pratt preprocessing function, as k (representing the rotation) will remain zero. Booth's algorithm runs in

$O(n)$

time, where n is the length of the string. The algorithm performs at most

$3n$

comparisons in the worst case, and requires auxiliary memory of length n to hold the failure function table.

Shiloach's Fast Canonization Algorithm[edit]

Shiloach (1981)[3] proposed an algorithm improving on Booth's result in terms of performance. It was observed that if there are ***q*** equivalent lexicographically minimal rotations of a string of length ***n***, then the string must consist of *q* equal substrings of length ***d=n/q***. The algorithm requires only ***n + d/2*** comparisons and constant space in the worst case.

The algorithm is divided into two phases. The first phase is a quick sieve which rules out indices that are obviously not starting locations for the lexicographically minimal rotation. The second phase then finds the lexicographically minimal rotation start index from the indices which remain.

Duval's Lyndon Factorization Algorithm[edit]

Duval (1983)[4] proposed an efficient algorithm involving the factorization of the string into its component Lyndon words, which runs in linear time with a constant memory requirement.

Variants[edit]

Shiloach (1979)[5] proposed an algorithm to efficiently compare two circular strings for equality without a normalization requirement. An additional application which arises from the algorithm is the fast generation of certain chemical structures without repetitions.