## NAME

sarray, ssarray, bsarray, lcp, scode − suffix-array tools

## SYNOPSIS

**#include "sarray.h"**
**int sarray(int *a, int n);**
**int ssarray(int *a);**
**int bsarray(unsigned char *s, int *a, int n);**
**int *lcp(int *a, char *s, int n);**
**unsigned char *scode(char *s);**

## DESCRIPTION

*Sarray* and *ssarray* convert array *a* into a suffix array for *a*. The *n* values in *a* must form a contiguous set of integers in the range 0 to some positive value, with 0 occurring only as an endmark, in *a*[*n*−1].

*Bsarray* builds, in *a* (of length *n*+1), a suffix array for the *n*-byte string *b*, which need not contain an endmark.

All three suffix-array builders return the index at which the whole string is identified in *a*. (This value is used in Burrows-Wheeler data compression.)

*Lcp* returns an array *l*, in which *l*[ *j*] is the length of the longest common prefix of the suffixes identified by *a*[*j*−1] and *a*[ *j*], except *l*[0]=0. It runs in time *O*(*n*) and uses temporary space equal in size to *a*.

*Scode* returns an encoding of string *s* in a form suitable for input to *sarray* or *ssarray.*

### Explanation

Suffix arrays are useful for information retrieval, biological sequence analysis, plagiarism detection, data compression, linguistic analysis, etc.

A suffix array identifies, in lexicographic order, the (positions of) the suffixes of a given string. Thus the suffix array for the string "abab", including its final null character, is {4,2,0,3,1}, identifying the suffixes "", "ab", "abab", "b", "bab". Equivalently, it identifies circular shifts in lexicographic order. For the string "abab", with # as a visible endmark, the shifts are "#abab", "ab#ab", "abab#", "b#aba", "bab#a".

The three array-building functions run in time *O*(*n* log *n*). *Sarray* and *bsarray* use a hybrid algorithm, typically several times as fast as the deliberately simple *ssarray* . All three require temporary space equal in size to *a.* Space overhead may be reduced by using *qsort*(3) with a suitable comparison function, but running time then becomes at best *O*(*nm* log *n*) *m* is the length of the longest repeated substring.

## EXAMPLES

Build, in *a* and *l* respectively, a suffix array for string *s* and the associated lcp array.

```
int *l;
int n = strlen(s)+1;
int *a = scode(s);
sarray(a, n);
l = lcp(a, s, n);
```

Build the same suffix array, using *bsarray.*

```
int n = strlen(s);
int *a = malloc((n+1)*sizeof(int));
bsarray((unsigned char*)s, a, n);
```

## DIAGNOSTICS

*Sarray, ssarray,* and *bsarray* return −1 for bad data or insufficient space.

*Lcp* and *scode* return *malloc*'ed arrays, or 0 for bad data or insufficient space.