```
COMPUTE-SUFFIX-ARRAY($T, n$)
allocate arrays $substr\text{-}rank[1:n]$, $rank[1:n]$, and $SA[1:n]$
for $i = 1$ to $n$
    $substr\text{-}rank[i].left\text{-}rank = \text{ord}(T[i])$
    if $i < n$
        $substr\text{-}rank[i].right\text{-}rank = \text{ord}(T[i+1])$
    else $substr\text{-}rank[i].right\text{-}rank = 0$
    $substr\text{-}rank[i].index = i$
sort the array substr-rank into monotonically increasing order based
    on the left-rank attributes, using the right-rank attributes to break ties;
    if still a tie, the order does not matter
$l = 2$
while $l < n$
    MAKE-RANKS($substr\text{-}rank, rank, n$)
    for $i = 1$ to $n$
        $substr\text{-}rank[i].left\text{-}rank = rank[i]$
        if $i + l \leq n$
            $substr\text{-}rank[i].right\text{-}rank = rank[i+l]$
        else $substr\text{-}rank[i].right\text{-}rank = 0$
        $substr\text{-}rank[i].index = i$
    sort the array substr-rank into monotonically increasing order based
        on the left-rank attributes, using the right-rank attributes
        to break ties; if still a tie, the order does not matter
    $l = 2l$
for $i = 1$ to $n$
    $SA[i] = substr\text{-}rank[i].index$
return $SA$
```

```
MAKE-RANKS($substr\text{-}rank, rank, n$)
$r = 1$
$rank[substr\text{-}rank[1].index] = r$
for $i = 2$ to $n$
    if $substr\text{-}rank[i].left\text{-}rank \neq substr\text{-}rank[i-1].left\text{-}rank$
            or $substr\text{-}rank[i].right\text{-}rank \neq substr\text{-}rank[i-1].right\text{-}rank$
        $r = r + 1$
    $rank[substr\text{-}rank[i].index] = r$
```