String Algorithms

Timothy Stranex

August 15, 2005

1 Notation

If S is the string 'abcdefghijkl' then,

- $S[0]$ is 'a' and $S[5]$ is 'f'.
- $S[0:5]$ is 'abcdef'.
- $S[0:5)$ is 'abcde'.
- $S(0:5]$ is 'bcdef'.
- $S(0:5)$ is 'bcde'.
- $S[5:0]$ is ''.
- $S(0:0)$ is ''.
- $S[:]$ is 'abcdefghijkl'.
- $S(:)$ is 'bcdefghijk'.
- $S[1:]$ is 'bcdefghijkl'.
- $S[:5]$ is 'abcdef'.
- $S(0:]$ is 'bcdefghijkl'.
- $S[:5)$ is 'abcde'.

2 Matching

Given $S[0:n)$ find the minimum i for which $S[i:i+m) = P[0:m)$.

2.1 Brute Force

```
for i = 0, 1, ··· , n − m + 1:
    found ← true
        for j = 0, 1, ··· , m − 1:
            if P[j] ≠ S[i + j]:
                found ← false
                break
        if found = true, terminate; P was found at position i.
Terminate; P was not found in S.
```

2.2 Finite State Machine

Constucting the failure function:

```
f(0) ← 0
f(1) ← 0
for i = 2, 3, ··· , m:
    j ← f(i − 1)
    while j > 0 and P[j] ≠ P[i − 1]:
```

```
        j ← f(j)
    if j = 0 and P[0] ≠ P[i − 1]:
        f(i) ← 0
    else:
        f(i) ← j + 1
```

Simulating the machine:

```
if m = 0, terminate; P occurs at position 0 in S.
if n = 0, terminate; P does not occur in S.
i ← 0
s ← 0
do:
    if S[i] = P[s]:
        s ← s + 1
        if s = m, terminate; P occurs at position i − m + 1 in S.
        i ← i + 1
    else if s = 0:
        i ← i + 1
    else:
        s = f(s)
while i < n
Terminate; P does not occur in S.
```

3 Longest Common Subsequence

4 Recursive

```
function lcs(A, B):
    if A =‘’ or B =‘’:
        return 0
    else if A[0] = B[0]:
        return 1 + lcs(A[1 :], B[1 :])
    else:
        return max(lcs(A[1 :], B), lcs(A, B[1 :]))
```

5 Dynamic Programming

```
function lcs(A, B):
    if length(B) < length(A):
        A ↔ B
    m ← length(A)
    n ← length(B)
    r1, r2 are arrays of size m + 1
    for i = n, n − 1, · · · , 0:
        for j = m, m − 1, · · · , 0:
            if j = m or i = n:
                r1[j] ← 0
            else if A[j] = B[i]:
                1[j] = 1 + r2[j + 1]
            else:
                r1[j] = max(r1[j + 1], r2[j])
        r1 ↔ r2
    return row2[0]
```