String Searching 1 of 2

String Searching

- Find the string "`vtewfvtxqwfczsrdzcaj`" in the following text:

```
qwerxcvvtewfzxcfasfedrsadfsdacfasdrtvtewqwertcsvte
wfvtxqwfczsrdzfeceeaeszxcvtsafsersdxzcvtedfaevsadv
tewfvtxqwfczsvzxgvtasfvtcasrfvtewqtrwtravtewfxtrac
wrtrdtgfdvxvvsbdgfstqtretydfxvzccadawqeewtertgfvbd
vczfafsvtewfvtxqwfczsgfsdfdxvzvzvtvsdgfsgtfwt6fqwt
qwrcfxtvtewfwtqwfzvwqgtfvtqfwcxetwfazreqresdqxrdqc
fwqdxvgfewcvtwefxvtrfczrqesxqecaqrfzvtqwxvbwyegcbe
bcwtfexvtfwxcrqxeqdcqzrwdfvtwxefvctyvtewfwefxqtfxc
qcdzrqxesrzqxrqcwqtfxtewfcvwerygcvewytxvqewtcxzdcd
qwfxvtewfvtxqwfczsrdzcajwfcsxtqwefdvetwqfvxdtqfwvq
```

Algorithms for string search

- string: S[0 .. m-1] ananaba
- text: T[0 .. n-1] bannabanabananaban

- Simple:

```
for i ← 0 to n-m-1
    found ← true
    for k ← 0 to m-1
        if S[k] != T[i+k] then found ← false, continue
    if found then return i
return -1
```

String search

- Simple search
  - Slide the window by 1
    - t = t +1;

abcdmndsjhhhsjgrjgslagfiigirnvkfir
abcdefg

ananfdfjoijtoiinkjjkjgfjgkjkkhgklhg
ananaba

- KMP
  - Why look at characters in the text multiple times?
  - Slide the window faster
    - t = t + s
    - but sometimes you can not skip s, need go back a little
    - a table to tell how to back step
    - t= t + s – M[s]

Knuth Morris Pratt

- string: S[0 .. m-1] ananaba
- text: T[0 .. n-1] anbanananananabnananaba
  ananaba

When there is a mismatch,

⇒ move the string along to the earliest place it could possibly match
and keep stepping

Need a table to say how far to match:

Is there a matching prefix of the match so far.

Match so far:	0	1	2	3	4	5	6
Move string along:	1	1	2	3-1	4-2	5-3	6
Next match from:	0	0	0	1	2	3	0

String search

- Simple search
  - Slide the window by 1
    - t = t +1;
- KMP
  - Slide the window faster
    - t = t + s – M[s]
  - Never recheck the matched characters
    - If there a “suffix ==prefix”?
      - No, skip these characters
        - » M[s] = 0
      - Yes, reuse, no need to recheck these characters
        - » M[s] is the length of the “reusable” suffix

abcdmndsjhhhsjgrjgslagfiigirnvkfir
abcdefg

ananfdfjoijtoiinkjjkjgfjgkjkkhgklhg
ananaba

Knuth Morris Pratt

```
input: string S[0 .. m-1] ,    text  T[0 .. n-1]
output:  the position in T at which S is found, or -1 if not present
variables:  s ← 0          position of current character in S
            t ← 0          start of current match in T
            M[0 .. m-1]    self match table
Construct self match table M
while   t + s   <  n
    if  S[ s ] = T[ t + s ]   then        //  match
        s ← s + 1
        if   s = m   then return  t       // found S
    else if M[ s ]  = -1   then           // mismatch, no self overlap
        s ← 0,    t ← t + s + 1,
    else                                  // mismatch, with self overlap
        t ← t + s - M[ s ]                // match position jumps forward
        s ← M[ s ]
return   -1      // failed to find S
```

KMP how far to move along?

- string: ananaba
- text: ...ananx???....
- If mismatch at string position s (and text position t+s)
  - find largest substring ending at s-1 that matches a prefix of string
  - move t to (t + s – length of substring)
  - keep matching from s ← length of substring
- special case:
  - if s = 0, then move t to t + 1 and match from s ← 0

KMP: Building the table.

```
input:    S[0 .. m-1]    // the string
output:  M[0 .. m-1]    // match table
initialise:  M[0] ← -1
             M[1] ← 0
             j ← 0            // position in prefix
             pos ← 2          // position in table
while pos < m
    if  S[pos - 1] = S[ j ]        //substrings  ...pos-1 and 0..j match
        M[pos] ←  j+1,
        pos++,   j++
    else if   j > 0                // mismatch, restart the prefix
        j ← M[ j ]
    else   // j = 0               // we have run out of candidate prefixes
        M[pos] ← 0,
        pos++
```

```
M:  0   1   2   3   4   5   6
    |   |   |   |   |   |   |
```

ananaba
ananaba

Knuth Morris Pratt

- Summary
  - searches forward,
  - never matches a text character twice (and never skips a text character)
  - jumps string forward based on self match within the string:
    - prefix of string matching a later substring.
    - doesn't use the character in the text to determine the jump.
- Cost?