MA/CSSE 473 Day 27

Student questions
Leftovers from Boyer-Moore

Knuth-Morris-Pratt String Search Algorithm

Solution (hide this until after class)

1. banana

k	shift
1	4
2	6
3	2
4	6
5	6

2. wowwow

k	shift
1	2
2	5
3	3
4	3
5	3

3. abcdcbcabcabc

k	shift
1	8
2	6
3	10
4	10
5	3
6	10
7	10
8	10
9	10
10	10
11	10
12	10

Boyer-Moore example (Levitin)

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	_
1	2	6	6	6	6	6	6	6	6	6	6	6	6	3	6	6	6	6	6	6	6	6	6	6	6	6

B E S S _ K N E W _ A B O U T _ B A O B A B S

B A O B A B

$d_1 = t_1(K) = 6$

B A O B A B

$d_1 = t_1(\_)-2 = 4$

$d_2(2) = 5$

B A O B A B

$d_1 = t_1(\_)-1 = 5$

$d_2(1) = 2$

B A O B A B (success)

k	pattern	d_2
1	BAOBAB	2
2	BAOBAB	5
3	BAOBAB	5
4	BAOBAB	5
5	BAOBAB	5

Boyer-Moore Example (mine)

```
pattern = abracadabra
text =
abracadabtabradabracadabcadaxbrabbracadabraxxxxxxabracadabracadabra
m = 11,  n = 67
badCharacterTable: a3 b2 r1 a3 c6 x11
GoodSuffixTable: (1,3) (2,10) (3,10) (4,7) (5,7) (6,7) (7,7) (8,7)
(9,7) (10, 7)

abracadabtabradabracadabcadaxbrabbracadabraxxxxxxabracadabracadabra
abracadabra
i =  10     k =  1     t1 =  11     d1 =  10     d2 =  3
abracadabtabradabracadabcadaxbrabbracadabraxxxxxxabracadabracadabra
          abracadabra
i =  20     k =  1     t1 =  6     d1 =  5     d2 =  3
abracadabtabradabracadabcadaxbrabbracadabraxxxxxxabracadabracadabra
               abracadabra
i =  25     k =  1     t1 =  6     d1 =  5     d2 =  3
abracadabtabradabracadabcadaxbrabbracadabraxxxxxxabracadabracadabra
                    abracadabra
i =  30     k =  0     t1 =  1     d1 =  1
```

Boyer-Moore Example (mine)

First step is a repeat from the previous slide

```
abracadabtabradabracadabcadaxbrabbracadabraxxxxxxabracadabracadabra
                    abracadabra
i =  30     k =  0     t1 =  1     d1 =  1
abracadabtabradabracadabcadaxbrabbracadabraxxxxxxabracadabracadabra
                     abracadabra
i =  31     k =  3     t1 =  11     d1 =  8     d2 =  10
abracadabtabradabracadabcadaxbrabbracadabraxxxxxxabracadabracadabra
                               abracadabra
i =  41     k =  0     t1 =  1     d1 =  1
abracadabtabradabracadabcadaxbrabbracadabraxxxxxxabracadabracadabra
                                abracadabra
i =  42     k =  10     t1 =  2     d1 =  1     d2 =  7
abracadabtabradabracadabcadaxbrabbracadabraxxxxxxabracadabracadabra
                                    abracadabra
i =  49     k =  1     t1 =  11     d1 =  10     d2 =  3
abracadabtabradabracadabcadaxbrabbracadabraxxxxxxabracadabracadabra
                                                abracadabra
49
```

Brute force took 50 times through the outer loop; Horspool took 13; Boyer-Moore 9 times.

Boyer-Moore Example

- On Moore's home page
- http://www.cs.utexas.edu/users/moore/best-ideas/string-searching/fstrpos-example.html

```
def search(pattern, text):
    n = len(text)
    m = len(pattern)
    badSymbolTable = [m]*128 # ASCII characters
    populateBadSymbolTable(badSymbolTable, pattern, m-1)
    goodSuffixTable = [m]*m
    populateGoodSuffixTable(pattern, goodSuffixTable)

    i = m -1
    while i < n:
         k = 0
        while k < m and pattern[m-1-k]==text[i-k]:
            k += 1;
        if k==m:
            return i - m + 1
        t1 = badSymbolTable[ord(text[i-k])]
        d1 = max(1, t1 - k)
         if k==0:
             i += d1
        else:
            i += max(d1, goodSuffixTable[k])
    return -1
```

This code is online

There is an O(m) algorithm for building the goodSuffixTable. It's complicated.

My code for building the table is $\Theta(m^2)$

This code is $\Theta(n)$

Knuth-Morris-Pratt Search

- Based on the brute force search.
- Does character-by-character matching left-to-right
- In many cases we can shift by more than 1, without missing any matches.
- Depends on repeated characters in p.
- We call the amount of the increment the *shift value*.
- Once we can calculate the correct shift values, the algorithm is fairly simple.
- Principles are like those behind Boyer-Moore Good Suffix shifts.

```
def search(pattern, text):
    n = len(text)
    m = len(pattern)
    table = [-1]*m # most of the -1's will be replaced.
    populateTable(table, pattern)

    i=0
    k=0
    comparisons = 0

    while i + k < n:
        comparisons += 1
        if pattern[k] == text[i + k]:
           if k == m - 1:
                return i
            k += 1
        elif table[k] >= 0:
            i = i + k - table[k]
            k = table[k]
       else:
            k = 0
            i += 1
    return -1
```

```
# build the table that tells how many characters matched.
def populateTable(table, pattern):
    m= len(pattern)
    table[1] = 0
    current = 2
    candidatePosition = 0

    while current < m:
        if pattern[current-1] == pattern[candidatePosition]:
            candidatePosition += 1
            table[current] = candidatePosition
            current += 1
        elif candidatePosition > 0:
            candidatePosition = table[candidatePosition]
        else:
            table[current] = 0
            current += 1
```

```
pattern = participate in parachute
text = apartyappisparticipating in partiesparticipate in parachuteparts
m = 24,  n = 64
[-1, 0, 0, 0, 0, 0, 0, 0, 1, 2, 0, 0, 0, 0, 0, 0, 1, 2, 3, 0, 0, 0, 0, 0]
apartyappisparticipating in partiesparticipate in parachuteparts
 participate in parachute    1
apartyappisparticipating in partiesparticipate in parachuteparts
     participate in parachute    5
apartyappisparticipating in partiesparticipate in parachuteparts
      participate in parachute    6
apartyappisparticipating in partiesparticipate in parachuteparts
       participate in parachute    7
apartyappisparticipating in partiesparticipate in parachuteparts
        participate in parachute    8
apartyappisparticipating in partiesparticipate in parachuteparts
         participate in parachute    9
apartyappisparticipating in partiesparticipate in parachuteparts
          participate in parachute    10
apartyappisparticipating in partiesparticipate in parachuteparts
           participate in parachute    11
apartyappisparticipating in partiesparticipate in parachuteparts
                     participate in parachute    21
apartyappisparticipating in partiesparticipate in parachuteparts
                      participate in parachute    22
```

```
apartyappisparticipating in partiesparticipate in parachuteparts
                        participate in parachute    23
apartyappisparticipating in partiesparticipate in parachuteparts
                         participate in parachute    24
apartyappisparticipating in partiesparticipate in parachuteparts
                          participate in parachute    25
apartyappisparticipating in partiesparticipate in parachuteparts
                           participate in parachute    26
apartyappisparticipating in partiesparticipate in parachuteparts
                     |       participate in parachute    27
apartyappisparticipating in partiesparticipate in parachuteparts
                             participate in parachute    28
apartyappisparticipating in partiesparticipate in parachuteparts
                                   participate in parachute    33
apartyappisparticipating in partiesparticipate in parachuteparts
                                    participate in parachute    34
apartyappisparticipating in partiesparticipate in parachuteparts
                                     participate in parachute    35
```