String Matching
A Lecture in CE Freshman Seminar Series:
Ten Puzzling Problems in Computer Engineering
About This Presentation

This presentation belongs to the lecture series entitled “Ten Puzzling Problems in Computer Engineering,” devised for a ten-week, one-unit, freshman seminar course by Behrooz Parhami, Professor of Computer Engineering at University of California, Santa Barbara. The material can be used freely in teaching and other educational settings. Unauthorized uses, including any use for financial gain, are prohibited. © Behrooz Parhami

<table>
<thead>
<tr>
<th>Edition</th>
<th>Released</th>
<th>Revised</th>
<th>Revised</th>
<th>Revised</th>
<th>Revised</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>May 2012</td>
<td>May 2015</td>
<td>May 2016</td>
<td>May 2020</td>
<td></td>
</tr>
</tbody>
</table>
Word Search Puzzles
Type 1, With Word List Supplied

AGITATOR
ASSEMBLY
CLUTCH
CONNECTORS
CONTROL
COUPLING
GLIDE
LINT SCREEN
PULLEY
SEAL
SWITCH
VALVE

The puzzle below is a little harder than the normal word search:
one of the 36 first/last names has been left out (which one?)

Amy Steel
Kevin Blair
Ron Palillo
Barbara Bingham
Kirsten Baker
Shavar Ross
Bruce Mahler
Larry Zerner
Stu Charno
Carol Lacatell
Mark Nelson
Susan Blu
Dana Kimmell
Paul Kratka
Tony Goldwyn
John Furey
Richard Young
Tracie Savage

May 2020
String Matching
Slide 3
“Ten Puzzling Problems in Computer Engineering” Word Search

WORD LIST:

BINARY SEARCH
BYZANTINE GENERALS
CRYPTOGRAPHY
EASY HARD IMPOSSIBLE
MALFUNCTION DIAGNOSIS
PLACEMENT AND ROUTING
SATISFIABILITY
SORTING NETWORKS
STRING MATCHING
TASK SCHEDULING

Puzzle generated at:
http://puzzlemaker.school.discovery.com/WordSearchSetupForm.html
Word Search Puzzle
Type 2, With Clues Supplied for the Words to be Found

Seven birds
☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐

Five units of length
☐ ☐ ☐ ☐ ☐

Four currencies
☐ ☐ ☐ ☐

Two things football players wear
☐ ☐

Large gland in the neck
☐


Q1: Solve the “Word Roundup” puzzle shown above

Q2: Build a “Word Roundup” puzzle where the clue is “15 country names.”
Converting a 2D Search to 1D Searches
A Challenging Hybrid Word Search Puzzle

```
E S A H C G T S I H D R E H T O
Y I E M U L E R O W G R P E N A
V N H A E B D A T O R L E E F N
A E R S L A N E A H A A E A G I
R D N I U I N S N T R T R M M
G Z N A A S E S K D G O E H P E
C G L R R N I A R T R E L R D A
R M D A J C S A Y B D A I S O S
A O N I O N C C D O R D O A S O
P E P O C H H C R R E M A B S A
S H C A E P M I E O D A I R T H
K D I M I T O W N N W I N O S C
```

CLUES

- ‘House, M.D.’s Robert
- “It makes its own _____”
- Dice game
- Basketball position
- Google “The”
- Unit of time & space
- Fuzzy fruit
- Shy
- Japanese food
- “_____ to an end”
- Whooping bird
- Rapper Gold
- Water goes down
- Wacky witches
- ‘White and _____’
- The Fifth Element
- Sawdust quiets
- Infinitesimals
- Deadly sin
- Deadly sin
- Deadly sin
- Deadly sin
- ‘Runaway _____’
- Two-by-four
- “Walk the _____”
- Dolce _____
- Drunks
- Ethan Rayne
- Japanese toons
- Peter Gabriel’s ‘Curtains’
- 1. This 2. That 3. _____
- ‘The Italian Job’s Seth’
Word-Search Puzzle with a Twist

This “Missing Peace Puzzle” was used in a qualifying round of World Puzzle Championships.

Supply the 16 missing letters at the center of the grid so that the word-search puzzle contains 18 of the 19 names of Nobel Peace Prize winners listed.

Q3: Solve the word-search puzzle above and supply the missing name.
String Matching: Problem Definition

Given a data string with $n$ symbols and a pattern string with $m$ symbols:

1. Does the pattern string appear in the data string?
2. What are the locations of all occurrences of the pattern in the data?

The brute-force, or sliding window, algorithm
Consider all possible positions where the pattern might begin ($n - m + 1$)
For each start position, do up to $m$ comparison to see if there is a match

Worst-case complexity = $O(mn)$; e.g., pattern “aaaaa”, data “aaaaaaaaaaaa”

Pattern string of length $m = 5$ symbols: EAGLE

Data string of length $n = 96$ symbols

May 2020
String Matching
Slide 9
Converting 2D Search Puzzles to 1D Searches

A 2D word search puzzle looks more exotic but it can be readily converted to a 1D string search puzzle.

Row-major order

<table>
<thead>
<tr>
<th>LEAGLEUROEXT</th>
<th>RA</th>
<th>X</th>
<th>W</th>
<th>Y</th>
<th>A</th>
<th>R</th>
<th>D</th>
<th>RX</th>
<th>EO</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>E</td>
<td>O</td>
<td>T</td>
<td>H</td>
<td>Y</td>
<td>R</td>
<td>O</td>
<td>I</td>
<td>D</td>
</tr>
<tr>
<td>TL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>N</td>
<td>S</td>
<td>N</td>
<td>T</td>
<td>E</td>
<td>T</td>
<td>P</td>
<td>B</td>
<td>N</td>
</tr>
<tr>
<td>AJCOZSLMOIMA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>Z</td>
<td>O</td>
<td>H</td>
<td>C</td>
<td>J</td>
<td>N</td>
<td>M</td>
<td>I</td>
<td>U</td>
</tr>
<tr>
<td>K</td>
<td>F</td>
<td>J</td>
<td>E</td>
<td>R</td>
<td>S</td>
<td>E</td>
<td>Y</td>
<td>E</td>
<td>L</td>
</tr>
<tr>
<td>VEGRETXZJTED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Column-major order

<table>
<thead>
<tr>
<th>LEAGLEUROEXT</th>
<th>RA</th>
<th>X</th>
<th>W</th>
<th>Y</th>
<th>A</th>
<th>R</th>
<th>D</th>
<th>RX</th>
<th>EO</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>E</td>
<td>O</td>
<td>T</td>
<td>H</td>
<td>Y</td>
<td>R</td>
<td>O</td>
<td>I</td>
<td>D</td>
</tr>
<tr>
<td>TL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>N</td>
<td>S</td>
<td>N</td>
<td>T</td>
<td>E</td>
<td>T</td>
<td>P</td>
<td>B</td>
<td>N</td>
</tr>
<tr>
<td>AJCOZSLMOIMA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>Z</td>
<td>O</td>
<td>H</td>
<td>C</td>
<td>J</td>
<td>N</td>
<td>M</td>
<td>I</td>
<td>U</td>
</tr>
<tr>
<td>K</td>
<td>F</td>
<td>J</td>
<td>E</td>
<td>R</td>
<td>S</td>
<td>E</td>
<td>Y</td>
<td>E</td>
<td>L</td>
</tr>
<tr>
<td>VEGRETXZJTED</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Insert a special symbol (#) between rows to ensure that new words or patterns are not created by the expansion.

Row-major order

| LEAGLEUROEXTRAXWYARDRXEOEOOTHYROIDTLHNSNTETPBANEL | AJCOZSLMOIMA | WZOHJCJMNIMIUNRKFJERSEYELNBVEGRETXZJTED |

Column-major order

| LEAGLEUROEXTRXWYARDRXEOEOOTHYROIDTLHNSNTETPBANEL | AJCOZSLMOIMA | WZOHJCJMNIMIUNRKFJERSEYELNBVEGRETXZJTED |

Similarly for (anti)diagonal.
Finding a Needle in a Haystack

Search for the 10-symbol “needle” h-e-l-e-n- -h-u-n-t in the Internet “haystack” with many TBs of data

The brute force algorithm amounts to: “Look in this corner, now in this other corner, then over there, and so on.”

The Internet holds some $1^+\text{ trillion}$ pages, growing by billions a day; each page on average contains in excess of 10 KB, say

$m \equiv 10$

$n \equiv 10^{12} \times 10^4 = 10^{16} \text{ B} = 10 \text{ PB} (\text{Petabyte}) = 0.01 \text{ EB} (\text{Exabyte})$

$\mathcal{O}(mn) \equiv 10^{17} \text{ comparisons} \Rightarrow 10^8 \text{ s} (>3 \text{ years}), \text{ with } 10^9 \text{ comparisons/s}$
Needle in a Haystack: Internet Search

Search for the 10-symbol string “helen hunt”

2.1M hits in 2009
5.4M hits in mid 2012
42.1M hits in 2016
Needle in a Haystack: Doing Less Work

For a particular pattern and unpredictable data strings, preprocess the pattern so that searching for it in various data strings becomes faster.

Analogy: Magnetize the needle

For a particular data string and unpredictable patterns, preprocess the data string so that when a pattern is supplied, we can readily find it with much less work.

Analogy: Do a thorough search of the haystack for different types of needles and place markers to guide future searches.
Example of Preprocessing the Pattern String

Devise an efficient method for finding the pattern “abcbab” in various data strings formed from the symbols a, b and c.

Data string:

```
 a b c b b b a b c b a b b c a a b c b a b c b b
```

O(n) instead of O(mn)
Example of Preprocessing the Data String

Devise an efficient method for finding various patterns in the data string:

```
abcbbbabbcabcbabccbaabcbb
```

Find all occurrences of the pattern “abcbab”

```
abc 0, 6, 15, 19, 23
bcb 1, 7, 16, 20, 24
cba 8, 17, 21
bab 5, 9, 18, 22
```

Q4: Use the index above to find the locations of babc in the string.
Another Preprocessing Example: Suffix Tree

A suffix tree is a tree in which root-to-leaf paths correspond to all the suffixes of a string.

Allows searching for a substring of length $m$ in $O(m)$ time instead of $O(n)$ time.

Example: Does the string "mississippi$" contain the substring "siss"? What about "pie"?

Image source:
http://docs.seqan.de/seqan/1.2/streeSentinel.png

Review article: CACM, April 2016, pp. 66-73.
Search Engine Indexes

17.5B hits in 2009
25.3B hits in 2012
25.3B hits in 2016

667 hits in 2009
1M+ hits in 2012
24.4K hits in 2016
Approximate String Matching

**Notion of string distance**
Each of the following transformations in a string creates a distance of 1

1. Insertion of an extra symbol
2. Deletion of a symbol
3. Transposition of two adjacent symbols

**Example distance-1 strings**
for *helen hunt*:

- *hellen hunt*
- *elen hunt*
- *helen hnut*

**Example distance-2 strings**
for *helen hunt*:

- *hellen hnut*
- *elen huntt*
- *lheen hunt*

Wildcard symbols can help in formulating approximate string searches

*h* *hunt* means any string that begins with an “*h*”, ends with “*hunt*”, and has an arbitrary set of symbols between the two

Melvyl (UC library catalog) allows such searches, e.g., author: *hunt h*
The (DNA) Sequence Alignment Problem

Given sequences S1 and S2 composed of the letters A, C, G, T
Determine their degree of similarity

S1: A G G G C T
S2: A G G C A

Application: Matching a given DNA sequence to a set of sequences in a database to find the best match

Dissimilarity arises from missing letters or mismatched letters

Alignment 1:
S1: A G G G C T
S2: A G G C A -

Penalty = GC mismatch + CA mismatch + 1 gap

Alignment 2:
S1: A G G G C T
S2: A G G - C A

Penalty = 1 gap + TA mismatch

Optimal alignment found via dynamic programming
A Couple of Bonus Word Search Puzzles

May 2020 String Matching Slide 20