# 122COM: Boyer-Moore Searching

Coventry University

# String searching better


- Naive search works but is inefficient.
- Obvious solution is not always the best one.
- Think about the problem and what is being searched.
  - Can you be smarter?

# Boyer-Moore

A

Boyer-Moore string searching algorithm.

- 1977.
- Not going to talk about the whole algorithm here.
  - Gets complex.

- Right to left comparison.
- Can skip sections of the text.
  - Don't need to test every position.
  - How?

# Boyer-Moore


Boyer-Moore string searching algorithm.

- 1977.
- Not going to talk about the whole algorithm here.
  - Gets complex.

- Right to left comparison.
- Can skip sections of the text.
  - Don't need to test every position.
  - How?
- Pre-processes the search string.
  - Produce bad character rule table.
  - Will explain how in a minute.

# Boyer-Moore algorithm

A

1. Preprocess the search string to create the “bad character table”.
   - Will explain how in a minute.

example ⇒

| a | e | l | m | p | x | * |
|---|---|---|---|---|---|---|
| 4 | 6 | 1 | 3 | 2 | 5 | 7 |

2. Same at the naive search, position the search string at the start of the main text.
3. Compare the strings.
4. If strings don’t match then in the bad character table lookup the character positioned at the end of the search string.
5. Move the search string by the number of positions specified in the table.
6. Repeat from step 3.

# Boyer-Moore II

A

example ⇒

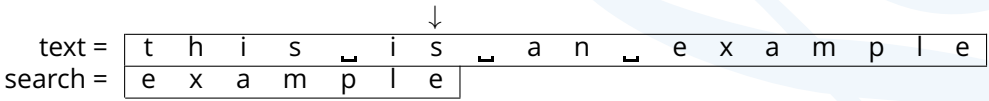
| a | e | l | m | p | x | * |
|---|---|---|---|---|---|---|
| 4 | 6 | 1 | 3 | 2 | 5 | 7 |

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| text = | t | h | i | s | ␣ | i | s | ␣ | a | n | ␣ | e | x | a | m | p | l | e |
| search = | e | x | a | m | p | l | e | | | | | | | | | | | |

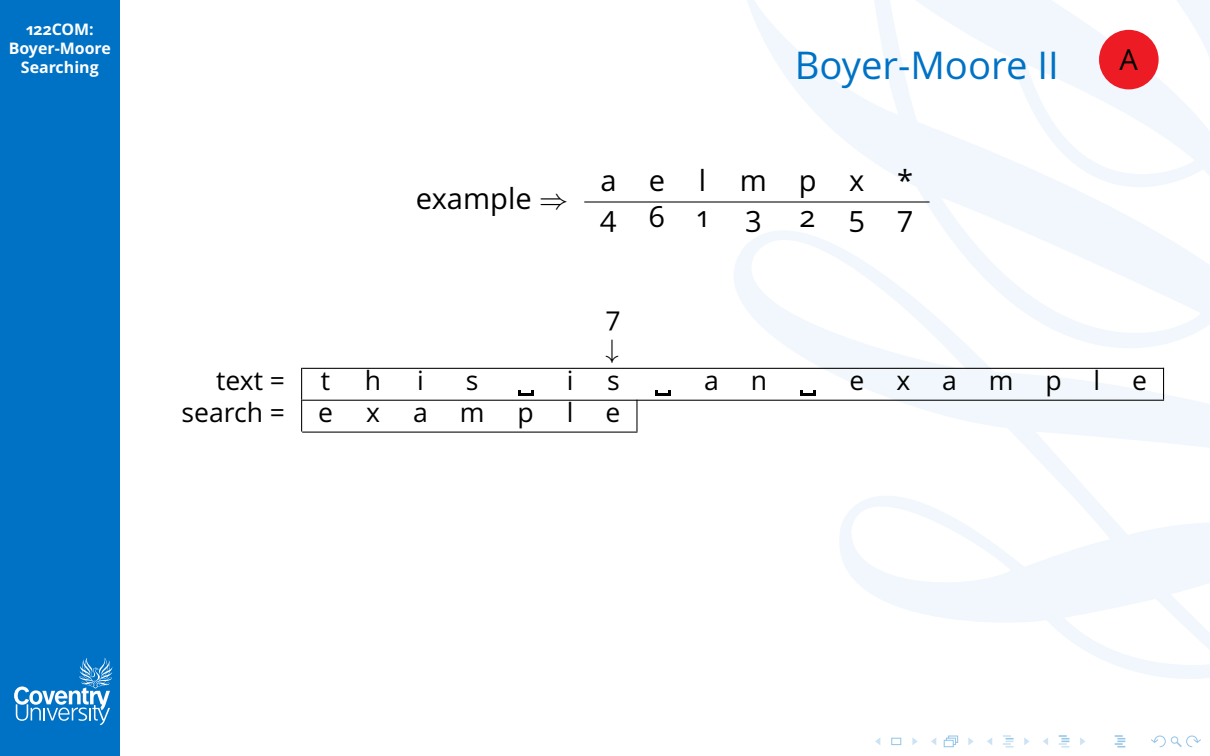
# Boyer-Moore II



| example ⇒ | a | e | l | m | p | x | * |
|---|---|---|---|---|---|---|---|
| | 4 | 6 | 1 | 3 | 2 | 5 | 7 |



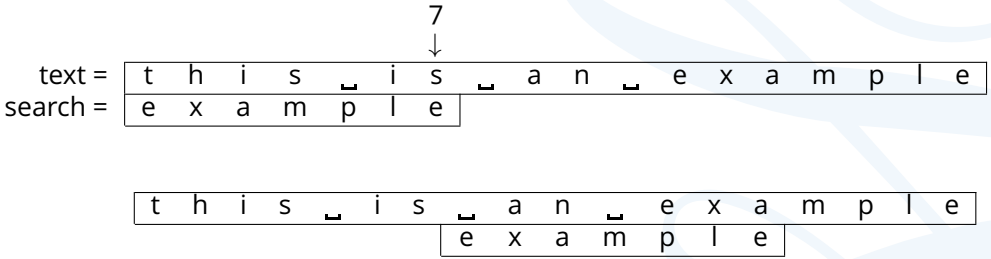
# Boyer-Moore II



# Boyer-Moore II



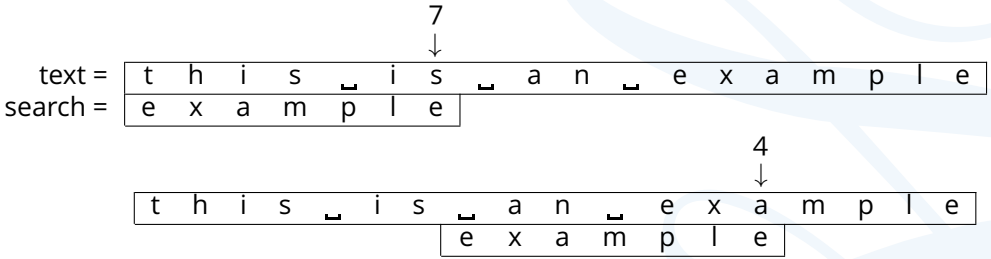
| example ⇒ | a | e | l | m | p | x | * |
|---|---|---|---|---|---|---|---|
| | 4 | 6 | 1 | 3 | 2 | 5 | 7 |



# Boyer-Moore II

A

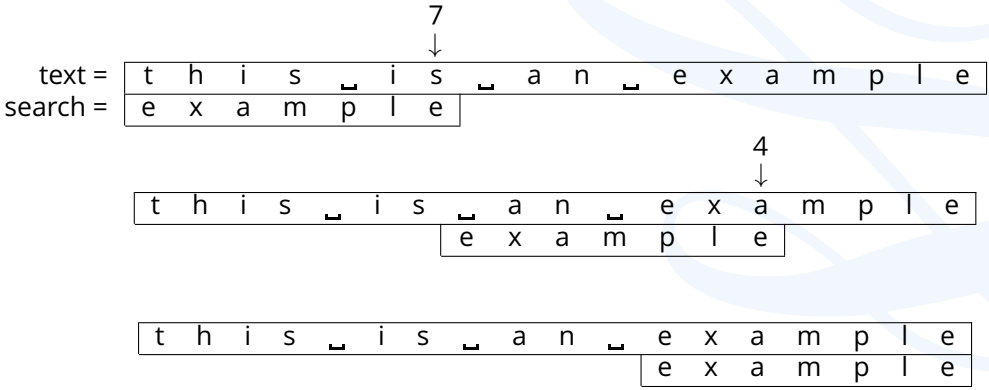
| example ⇒ | a | e | l | m | p | x | * |
|---|---|---|---|---|---|---|---|
| | 4 | 6 | 1 | 3 | 2 | 5 | 7 |



# Boyer-Moore II

A

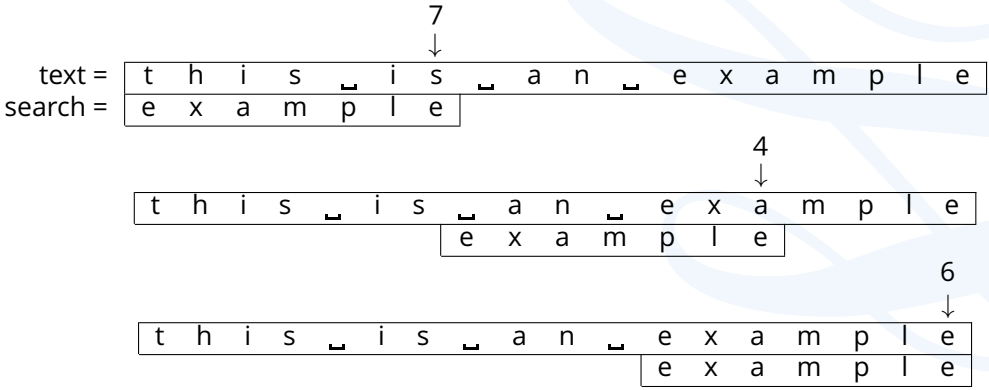
| example ⇒ | a | e | l | m | p | x | * |
|---|---|---|---|---|---|---|---|
| | 4 | 6 | 1 | 3 | 2 | 5 | 7 |



# Boyer-Moore II

A

| example ⇒ | a | e | l | m | p | x | * |
|---|---|---|---|---|---|---|---|
| | 4 | 6 | 1 | 3 | 2 | 5 | 7 |



# Boyer-Moore III


Creating the bad character table.

- For each character.
  - Except the last.
- Just count number of places between it and end of search string.

example $\Rightarrow$

| a | e | l | m | p | x | * |
|---|---|---|---|---|---|---|

# Boyer-Moore III


Creating the bad character table.

- For each character.
  - Except the last.
- Just count number of places between it and end of search string.

example ⇒

| a | e | l | m | p | x | * |
|---|---|---|---|---|---|---|
| 4 | | | | | | |

# Boyer-Moore III


Creating the bad character table.

- For each character.
  - Except the last.
- Just count number of places between it and end of search string.

example $\Rightarrow$

| a | e | l | m | p | x | * |
|---|---|---|---|---|---|---|
| 4 | 6 | | | | | |

# Boyer-Moore III


Creating the bad character table.

- For each character.
  - Except the last.
- Just count number of places between it and end of search string.

example ⇒

| a | e | l | m | p | x | * |
|---|---|---|---|---|---|---|
| 4 | 6 | 1 | | | | |

# Boyer-Moore III


Creating the bad character table.

- For each character.
  - Except the last.
- Just count number of places between it and end of search string.

example ⇒

| a | e | l | m | p | x | * |
|---|---|---|---|---|---|---|
| 4 | 6 | 1 | 3 | | | |

# Boyer-Moore III


Creating the bad character table.

- For each character.
  - Except the last.
- Just count number of places between it and end of search string.

example ⇒

| a | e | l | m | p | x | * |
|---|---|---|---|---|---|---|
| 4 | 6 | 1 | 3 | 2 | | |

# Boyer-Moore III


Creating the bad character table.

- For each character.
  - Except the last.
- Just count number of places between it and end of search string.

example ⇒

| a | e | l | m | p | x | * |
|---|---|---|---|---|---|---|
| 4 | 6 | 1 | 3 | 2 | 5 | |

# Boyer-Moore III


Creating the bad character table.

- For each character.
  - Except the last.
- Just count number of places between it and end of search string.

example ⇒

| a | e | l | m | p | x | * |
|---|---|---|---|---|---|---|
| 4 | 6 | 1 | 3 | 2 | 5 | 7 |

# Boyer-Moore IV

A

Doesn't need to sort or modify the sequence being searched.

- Small amount of pre-processing on the search value.

Worst case.

- Linear time.

Average case

- Sub-linear.

Not the only improved string searching algorithm.

- Knuth-Morris-Pratt.
- Finite State Machine (FSM).
- Rabin-Karp.

# The End