comp10002
Foundations of Algorithms

Semester Two, 2025

Strings and Algorithms

© The University of Melbourne, 2025
Lecture slides prepared by Alistair Moffat

Overview

Chapter 7 (Part II)

Pattern search

KMP pattern search

BMH pattern search

String index structures

Chapter 7 – Concepts (Part II)

There is no pre-defined string type in C, and they are stored as null-terminated arrays of characters.

String operations are carried out using character pointers.

The libraries `ctype.h` and `string.h` contain useful functions, such as `isalpha()` and `strcmp()`.

The function `malloc()` (Chapter 10, coming soon in the lec07 lecture slides) can be used to create space for new or resized arrays, and hence new or lengthened strings.

Chapter 7 – Program examples (Part II)

- ▶ string1.c
- ▶ strcpy.c
- ▶ getword.c
- ▶ words.c
- ▶ progargs.c

comp10002
Foundations of
Algorithms

lec06

Chapter 7
Pattern search
KMP search
BMH search
Indexing

You may not modify either of the two strings in the first three exercises.

Exercise 1

Write a function `is_subsequence(char *s1, char *s2)` that returns `1` if the characters in `s1` appear within `s2` in the same order as they appear in `s1`. For example, `is_subsequence("bee", "abbreviate")` should be `1`, whereas `is_subsequence("bee", "acerbate")` should be `0`.

Chapter 7 – Exercises

comp10002
Foundations of
Algorithms

lec06

Chapter 7
Pattern search
KMP search
BMH search
Indexing

Exercise 2

Ditto arguments, but determining whether every occurrence of a character in `s1` also appears in `s2`, and `0` otherwise. For example, `is_subset("bee", "rebel")` should be `1`, whereas `is_subset("bee", "brake")` should be `0`.

Exercise 3

Write a function `is_anagram(char *s1, char *s2)` that returns `1` if the two strings contain the same letters, possibly in a different order, and `0` otherwise, ignoring whitespace characters, and ignoring case. For example, `is_anagram("Algorithms", "Glamor Hits")` should return `1`.

Chapter 7 – Exercises

Exercise 4

Write a function `next_perm(char *s)` that rearranges the characters in a string argument and generates the lexicographically next permutation of the same letters. For example, if the string `s` is initially `"51432"`, then when the function returns `s` should be `"52134"`.

Exercise 5

If the two strings are of length n (and, if there are two, m), what is the asymptotic performance of your answers to Exercises 1–4?

Chapter 7 – Summary (Part II)

Key messages:

▶ Strings are stored in character arrays

▶ The underlying array must be declared big enough to hold the string plus a sentinel byte

▶ Functions to manipulate strings inevitably make use of `char*` pointers

▶ Arrays of `char*` are used to manipulate sets of strings, including `argv`, the initiating command line.

Pattern search

comp10002
Foundations of
Algorithms

lec06

Chapter 7

Pattern search

KMP search

BMH search

Indexing

Given: A text sequence $T[0 \ldots n-1]$ and a pattern $P[0 \ldots m-1]$.

Question: Does pattern P appear as a continuous subsequence of text T? If so, where?

Sequential pattern search

comp10002
Foundations of
Algorithms

lec06

Chapter 7

Pattern search
KMP search
BMH search
Indexing

$s \leftarrow 0$

while $s \leq n - m$

    for $i \leftarrow 0$ **to** $m - 1$

        if $T[s + i] \neq P[i]$

            break

    if $i = m$

        return s

    $s \leftarrow s + 1$

return *not_found*

Sequential pattern search

comp10002
Foundations of
Algorithms

lec06

Chapter 7

Pattern search

KMP search

BMH search

Indexing

Running time?

In the worst case (what inputs?), requires $O(nm)$ time.

Average case (but remember, need to be careful with this!) is $O(n)$.
Is linear time worst case possible? Or even sub-linear?

KMP pattern search

comp10002
Foundations of
Algorithms

lec06

Chapter 7

Pattern search

KMP search

BMH search

Indexing

Great idea: Start with a standard first alignment, and extend a match as far as possible. If/when a mismatch occurs, shift the pattern forward as far as possible without moving past any matching prefix of the pattern.

In the cases where pattern shifts forward by less than i (the current position in P), the new i gets set accordingly.

The search location in T described by $s + i$ never moves backwards.

KMP – Example

comp10002
Foundations of
Algorithms

lec06

Chapter 7

Pattern search

KMP search

BMH search

Indexing

Example: does `she shells` appear in `she sells sea shells`.

```
0         0         1         1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
s h e   s e l l s   s e a   s h e l l s
s h e   s h e l l s
```

Variables: $s = 0$ and $i = 0$.

KMP – Example

comp10002
Foundations of
Algorithms

lec06

Chapter 7

Pattern search

KMP search

BMH search

Indexing

Example: does `she shells` appear in `she sells sea shells`.

```
0           0           1           1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
s h e   s e l l s   s e a   s h e l l s
s h e   s h e l l s
* * * * * X
```

Variables: $s = 0$ and $i = 5$. Mismatch

KMP – Example

comp10002
Foundations of
Algorithms

lec06

Chapter 7

Pattern search

KMP search

BMH search

Indexing

Example: does `she shells` appear in `she sells sea shlls`.

```
0           0           1           1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9

s h e   s e l l s   s e a   s h e l l s
      s h e   s h e l l s
```

Variables: $s = 4$ and $i = 1$.

KMP – Example

comp10002
Foundations of
Algorithms

lec06

Chapter 7

Pattern search

KMP search

BMH search

Indexing

Example: does `she shells` appear in `she sells sea shells`.

```
0           0           1           1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9

s h e   s e l l s   s e a   s h e l l s
        s h e   s h e l l s
        - X
```

Variables: $s = 4$ and $i = 1$. Mismatch

KMP – Example

comp10002
Foundations of
Algorithms

lec06

Chapter 7

Pattern search

KMP search

BMH search

Indexing

Example: does `she shells` appear in `she sells sea shells`.

```
0           0           1           1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
s h e   s e l l s   s e a   s h e l l s
        s h e   s h e l l s
```

Variables: $s = 5$ and $i = 0$.

KMP – Example

comp10002
Foundations of
Algorithms

lec06

Chapter 7

Pattern search

KMP search

BMH search

Indexing

Example: does `she shells` appear in `she sells sea shells`.

```
0           0           1           1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
s h e   s e l l s   s e a   s h e l l s
        s h e   s h e l l s
        X
```

Variables: $s = 5$ and $i = 0$. Mismatch

KMP – Example

comp10002
Foundations of
Algorithms

lec06

Chapter 7

Pattern search

KMP search

BMH search

Indexing

Example: does `she shells` appear in `she sells sea shells`.

```
0           0           1           1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
s h e   s e l l s   s e a   s h e l l s
          s h e   s h e l l s
```

Variables: $s = 6$ and $i = 0$.

Etc.

KMP pattern search

comp10002
Foundations of
Algorithms

lec06

Chapter 7
Pattern search
KMP search
BMH search
Indexing

Define $F[i]$ to be the maximum $k < i$ such that $P[0 \ldots k-1]$ matches $P[i-k \ldots i-1]$, with $F[0]$ set to be -1.

Then at each mismatch, can shift P right by i (mismatch position) minus $F[i]$ (allowance for pattern self-overlap).

If $F[i]$ is zero (common case), then pattern search resumes from mismatched location $s + i$, rather than $s + 1$.

Cool!

KMP pattern matching

Examples of F:

```
      0           0             1             1
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5

P:  s h e   s h e l l s
F: -1 0 0 0 0 1 2 3 0 0

P:  s h e   s e l l s   s h e l l s
F: -1 0 0 0 0 1 0 0 0 1 0 1 2 3 0 0

P:  a a a a a a a a
F: -1 0 1 2 3 4 5 6

P:  a b c d a b c d a b c d e f g
F: -1 0 0 0 0 1 2 3 4 5 6 7 8 0 0
```

KMP pattern matching

comp10002
Foundations of
Algorithms

lec06

Chapter 7

Pattern search

KMP search

BMH search

Indexing

With F created, doing the search is easy:

```
s, i ← 0, 0
while s ≤ n − m
    if T[s + i] = P[i]
        i ← i + 1
        if i = m
            return s
    else
        s ← s + i − F[i]
        i ← max(F[i], 0)
return not_found
```

KMP pattern search

comp10002
Foundations of
Algorithms

lec06

Chapter 7
Pattern search
KMP search
BMH search
Indexing

Building the failure function F for pattern $P[0 \ldots m-1]$ makes use of very similar logic:

$s, c \leftarrow 2, 0$

$F[0], F[1] \leftarrow -1, 0$

while $s < m$

    if $P[c] = P[s-1]$

        $c, F[s], s \leftarrow c+1, c+1, s+1$

    else if $c > 0$

        $c \leftarrow F[c]$

    else

        $F[s], s \leftarrow 0, s+1$

KMP pattern search

comp10002
Foundations of
Algorithms

lec06

Chapter 7

Pattern search

KMP search

BMH search

Indexing

Example: `she shells`.

```
    0         0           1             1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
P:  s h e     s h e l l s
    *     *
F: -1 0
```

Variables: $c = 0$, $s = 2$: $P[c] \neq P[s - 1]$,
  so $F[2] \leftarrow 0$

KMP pattern search

comp10002
Foundations of
Algorithms

lec06

Chapter 7
Pattern search
KMP search
BMH search
Indexing

Example: `she shells`.

```
    0         0         1         1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5

P:  s h e   s h e l l s
    *       *
F: -1 0 0
```

Variables: $c = 0$, $s = 3$: $P[c] \neq P[s-1]$,
    so $F[3] \leftarrow 0$

KMP pattern search

comp10002
Foundations of
Algorithms

lec06

Chapter 7
Pattern search
KMP search
BMH search
Indexing

Example: `she shells`.

```
    0         0          1          1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5

P:  s h e   s h e l l s
    *       *
F: -1 0 0 0
```

Variables: $c = 0$, $s = 4$: $P[c] \neq P[s - 1]$,
    so $F[4] \leftarrow 0$

KMP pattern search

Example: `she shells`.

```
      0         0         1         1
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5

P:    s h e   s h e l l s
      *         *
F: -1 0 0 0 0
```

Variables: $c = 0$, $s = 5$: $P[c] = P[s - 1]$,
   so $F[5] \leftarrow 1$ and $c \leftarrow 1$

KMP pattern search

comp10002
Foundations of
Algorithms

lec06

Chapter 7
Pattern search
KMP search
BMH search
Indexing

Example: `she shells`.

```
    0         0         1         1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5

P:  s h e   s h e l l s
      *       *
F: -1 0 0 0 0 1
```

Variables: $c = 1$, $s = 6$: $P[c] = P[s-1]$,
  so $F[6] \leftarrow 2$ and $c \leftarrow 2$

KMP pattern search

Example: `she shells`.

```
    0         0         1         1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5

P:  s h e   s h e l l s
          *           *
F: -1 0 0 0 0 1 2
```

Variables: $c = 2$, $s = 7$: $P[c] = P[s-1]$,
    so $F[7] \leftarrow 3$ and $c \leftarrow 3$

KMP pattern search

comp10002
Foundations of
Algorithms

lec06

Chapter 7
Pattern search
KMP search
BMH search
Indexing

Example: `she shells`.

```
    0         0           1           1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
P:  s h e   s h e l l s
          *           *
F: -1 0 0 0 0 1 2 3
```

Variables: $c = 3$, $s = 8$: $P[c] \neq P[s-1]$,
    so $c \leftarrow F[3]$

KMP pattern search

Example: `she shells`.

```
      0           0           1           1
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5

P:  s h e   s h e l l s
    *                   *
F: -1 0 0 0 0 1 2 3
```

Variables: $c = 0$, $s = 8$: $P[c] \neq P[s-1]$,
    so $F[8] \leftarrow 0$

KMP pattern search

comp10002
Foundations of
Algorithms

lec06

Chapter 7

Pattern search

KMP search

BMH search

Indexing

Example: `she shells`.

```
    0         0         1         1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5

P:  s h e   s h e l l s
    *                 *
F: -1 0 0 0 0 1 2 3 0
```

Variables: $c = 0$, $s = 9$: $P[c] \neq P[s-1]$,
    so $F[9] \leftarrow 0$

KMP pattern search

comp10002
Foundations of
Algorithms

lec06

Chapter 7
Pattern search
KMP search
BMH search
Indexing

Example: `she shells`.

```
    0         0           1           1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
P:  s h e   s h e l l s
    *                   *
F: -1 0 0 0 0 1 2 3 0 0
```

Variables: $c = 0$, $s = 10$: $P[c] = P[s-1]$,
    so $F[10] \leftarrow 0$ and $c \leftarrow 1$

KMP – Analysis

comp10002
Foundations of
Algorithms

lec06

Chapter 7
Pattern search
KMP search
BMH search
Indexing

Named after Knuth, Morris, Pratt, who invented it in 1974.

Analysis? In main search loop, at every iteration, either:

- i goes up by one and s is unchanged; or
- s goes up by the same as i decreases; or
- s goes up by 1 and i remains zero.

In all three cases the quantity $2s + i$ increases by at least one.

But since $s < n$ and $i \leq m$, the number of loop iterations before exit with either success or failure is at most $2n + m$.

KMP – Analysis

comp10002
Foundations of
Algorithms

lec06

Chapter 7
Pattern search
KMP search
BMH search
Indexing

A similar argument applies to the construction of F:

- s and c both go up by one; or
- c decreases by at least one; or
- s goes up by one and c remains zero.

In all cases, $2s - c$ increases by at least one.

But $c > 0$ and $s < m$; hence the total number of iterations is less than $2m$.

The preprocessing phase does not dominate.

Boyer-Moore-Horspool pattern search

comp10002
Foundations of
Algorithms

lec06

Chapter 7
Pattern search
KMP search
BMH search
Indexing

Another clever idea: Start from the right-hand end of the pattern, and work to the left.

For each symbol v in the input alphabet, define $L[v]$ to be the shift needed to bring the rightmost location in the pattern at which v appears into the place in the text where the pattern previously ended.

If/when a mismatch occurs, the pattern can be shifted right by $L[T[s + m - 1]]$ to force last character to be in alignment.

BMH – Example

```
0           0           1           1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
s h e   s e l l s   s e a   s h e l l s
s h e   s h e l l s
```

Variables: $s = 0$ and $i = 9$.

BMH – Example

comp10002
Foundations of
Algorithms

lec06

Chapter 7

Pattern search

KMP search

BMH search

Indexing

```
0           0           1           1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
s h e   s e l l s   s e a   s h e l l s
s h e   s h e l l s
                X
```

Variables: $s = 0$ and $i = 9$.   Mismatch.

BMH – Example

comp10002
Foundations of
Algorithms

lec06

Chapter 7
Pattern search
KMP search
BMH search
Indexing

```
0         0         1         1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9

s h e   s e l l s   s e a   s h e l l s
          s h e   s h e l l s
```

Variables: $s = 6$ and $i = 9$.

BMH – Example

comp10002
Foundations of
Algorithms

lec06

Chapter 7

Pattern search

KMP search

BMH search

Indexing

```
0           0           1           1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9

s h e   s e l l s   s e a   s h e l l s
          s h e   s h e l l s
                            X
```

Variables: $s = 6$ and $i = 9$.   Mismatch.

BMH – Example

comp10002
Foundations of
Algorithms

lec06

Chapter 7
Pattern search
KMP search
BMH search
Indexing

```
0           0           1           1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9

s h e   s e l l s   s e a   s h e l l s
                    s h e   s h e l l s
```

Variables: $s = 10$ and $i = 9$.

BMH – Example

comp10002
Foundations of
Algorithms

lec06

Chapter 7

Pattern search

KMP search

BMH search

Indexing

```
0           0           1           1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9

s h e   s e l l s   s e a   s h e l l s
                    s h e   s h e l l s
                    X * * * * * * *
```

Variables: $s = 10$ and $i = 2$.   Mismatch.

BMH – Example

comp10002
Foundations of
Algorithms

lec06

Chapter 7

Pattern search

KMP search

BMH search

Indexing

```
0         0         1         1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
s h e   s e l l s   s e a   s h e l l s
                        s h e   s h e l l s
```

Variables: $s = 15$ and $i = 9$ and $s + i > 19$.

End of search. Only 10 character comparisons were done!

BMH – Shift array

Construct the shift array L for pattern length m and alphabet $0 \ldots \sigma - 1$:

for $v \leftarrow 0$ **to** $\sigma - 1$
    $L[v] \leftarrow m$
for $i \leftarrow 0$ **to** $m - 2$
    $L[P[i]] = m - i - 1$

BMH – Shift array

Examples of V:

```
     0         0         1         1
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5

P:   s h e   s h e l l s
L:   (s,5), (h,4), (e,3), ( ,6), (l,1), (a,10)

P:   s h e   s e l l s   s h e l l s
L:   (s,5), (h,4), (e,3), ( ,6), (l,1), (a,16)

P:   a a a a a a a a
L:   (a,1), (b,8), (c,8)

P:   a b c d a b c d a b c d e f g
L:   (a,6), (b,5), (c,4), (d,3), (e,2), (f,1), (g,15)
```

BMH – Searching

comp10002
Foundations of
Algorithms

lec06

Chapter 7

Pattern search

KMP search

BMH search

Indexing

Search the string:

$s, i \leftarrow 0, m - 1$

while $s \leq n - m$

    if $T[s + i] \neq P[i]$

        $s, i \leftarrow s + L[T[s + m - 1]], m - 1$

    else if $i = 0$

        return s

    else

        $i \leftarrow i - 1$

return not_found

BMH – Analysis

comp10002
Foundations of
Algorithms

lec06

Chapter 7
Pattern search
KMP search
BMH search
Indexing

In the worst case, back up to $O(nm)$, and not interesting.

But average case is much better, and experimentally is very fast for large alphabets (ASCII) and shortish patterns (m under 10 or so), because it can leapfrog quickly down T, looking at only a small number of characters at each leap.

Note that "average" must be from input data; these is no sense in which randomness can be introduced in to algorithm.

BMH – Variations

comp10002
Foundations of
Algorithms

lec06

Chapter 7
Pattern search
KMP search
BMH search
Indexing

Plenty of extensions have been proposed:

- ▶ Scan from left to right, to get better cache/prefetch behavior;
- ▶ Use two final characters, shift by larger of two;
- ▶ Use a full $m \times \sigma$ array, so that shift amount depends on position as well as missed character;
- ▶ Use two final characters as a combination, shift by full amount m if that adjacent pair does not appear earlier in the string;
- ▶ Take into account the part of the suffix that has been matched, KMP-style.

Indexing

comp10002
Foundations of
Algorithms

lec06

Chapter 7

Pattern search

KMP search

BMH search

Indexing

What happens if T is fixed and large ($m \ll n$), and there are going to be multiple independent patterns to be checked?

Is there some way of precomputing an index?

Of course there is, lots of choices. . .

Suffix array indexing

comp10002
Foundations of
Algorithms

lec06

Chapter 7
Pattern search
KMP search
BMH search
Indexing

Consider text $T[0 \ldots n-1]$.

Suppose that $T[n] = \$$, a unique symbol smaller than any other symbol.

Define $T_i = T[i \ldots n]$ to be the ith suffix of T.

A suffix array $S[0 \ldots n-1]$ is an array of pointers $S[i]$ such that $T_{S[i]}$ lexicographically precedes $T_{S[i+1]}$.

Suffix array – Example

comp10002
Foundations of
Algorithms

lec06

Chapter 7

Pattern search

KMP search

BMH search

Indexing

```
        0           0           1           1
        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7
        s h e # s e l l s # s h e l l s
```

i	$S[i]$	$T_{S[i]}$
0	16	$
1	3	# s e l l s # s h e l l s $
2	9	# s h e l l s $
3	2	e # s e l l s # s h e l l s $
4	12	e l l s $
5	5	e l l s # s h e l l s $
6	1	h e # s e l l s # s h e l l s $
7	11	h e l l s $
8	13	l l s $
9	6	l l s # s h e l l s $
10	14	l s $
11	7	l s # s h e l l s $
12	15	s $
13	8	s # s h e l l s $
14	4	s e l l s # s h e l l s $
15	0	s h e # s e l l s # s h e l l s $
16	10	s h e l l s $

Suffix array pattern search

Looking for a pattern $P[0 \ldots m-1]$?

Use binary search in S, comparing the pattern P against the suffixes in T, examining as long a prefix of
$T_{S[i]} = T[S[i] \ldots \min\{S[i] + m, n\}]$ as is necessary in each comparison, to identify a range of matches in S.

Takes $O(\log n)$ string comparisons via S. Each string comparison takes at most m character comparisons.

So total time is $O(m \log n)$ per search. That's very fast!

Remember this problem?

comp10002
Foundations of
Algorithms

lec06

Chapter 7

Pattern search

KMP search

BMH search

Indexing

Exercise 6

Given: A sequence S of n symbols

Problem: Find all locations in S at which repeated subsequences of length m or more appear.

Would a suffix array be useful??

Suffix array construction

comp10002
Foundations of
Algorithms

lec06

Chapter 7

Pattern search

KMP search

BMH search

Indexing

Just one small problem: generating the suffix array.

Simple approach: use an $O(n \log n)$-comparison sorting algorithm, with each comparison requiring $O(\log n)$ steps on average, and as many as n steps in the worst case.

Overall, $O(n \log^2 n)$ time average case on random strings, and $O(n^3)$ worst case if we use quicksort and have a "bad" string. Might not be cheap!

Note that Chapter 12 introduces $O(n \log n)$ worst-case sorting methods. That takes the worst-case for suffix array construction to $O(n^2 \log n)$.

Sorting strings

comp10002
Foundations of
Algorithms

lec06

Chapter 7

Pattern search

KMP search

BMH search

Indexing

Ternary Quicksort: partition on one character, at depth d in the strings. Then do three recursive calls, advancing to depth $d + 1$ in the "equals" section.

To create a suffix array for a string T:

> $tquicksort(S, n, d)$:
>
> > $p \leftarrow T[S[i] + d]$ for some $0 \leq i < n$
> > $(fe, fg) \leftarrow partition(S, n, p, d)$
> > $tquicksort(S, fe, d)$
> > $tquicksort(S + fe, fg - fe, d + 1)$
> > $tquicksort(S + fg, n - fg, d)$
>
> Initial call:
> $tquicksort(S, n, 0)$

Sorting strings

comp10002
Foundations of
Algorithms

lec06

Chapter 7
Pattern search
KMP search
BMH search
Indexing

Analysis: tricky. But, roughly speaking, shaves a factor of between log n (average case) to n (worst case) off execution time.

Worst case drops from $O(n^3)$ to $O(n^2)$. Average case analysis still requires randomness in the data.

Experimentally, works well on typical non-pathological texts.

Suffix array construction is an active area of algorithmic research. The best current methods take $O(n)$ time, but too much space to be fully practical. There will probably be new algorithms five years from now.

Indexing – Exercises

Exercise 7:

A KWIC index for a text (KeyWord In Context) presents a small window of words around each (case folded) word that appears in the text, in dictionary order.

Write a program that outputs a KWIC Index for the text that is provided as input. Only whole words should be indexed.

You may assume that at most 10,000 words will be input.

Indexing – Exercises

For example, for the string

```
She sells sea shells by the sea shore
```

and for a window size of two words either side, the output of the indexing program should be:

```
    sea shells *by the sea
     She sells *sea shells by
        by the *sea shore
           She *sells sea shells
               *She sells sea
   sells sea *shells by the
      the sea *shore
    shells by *the sea shore
```

Indexing – Exercises

comp10002
Foundations of
Algorithms

lec06

Chapter 7

Pattern search

KMP search

BMH search

Indexing

Exercise 8:

An inverted index for a text is an alphabetical listing of all of the words that appear, together with the line numbers(s) at which they appear:

Write a program that generates an inverted index for the text that is provided as input. Words should be case-folded.

You may assume that at most 10,000 words will be input.

Indexing – Exercises

```
mac:./inverted_index
She sells sea shells
by the sea shore
He sells sea shells too
sells to see her more
^D
by      : 2
he      : 3
her     : 4
more    : 4
sea     : 1, 2, 3
see     : 4
sells   : 1, 3, 4
she     : 1
shells  : 1, 3
shore   : 2
the     : 2
to      : 4
too     : 3
```