https://algs4.cs.princeton.edu

5.1 String Sorts

- strings in Java
- key-indexed counting
- LSD radix sort
- MSD radix sort
- 3-way radix quicksort
- suffix arrays

Keyword-in-context search

Given a text of n characters, preprocess it to enable fast substring search (find all occurrences of query string context).

```
% more tale.txt
it was the best of times
it was the worst of times
it was the age of wisdom
it was the age of foolishness
it was the epoch of belief
it was the epoch of incredulity
it was the season of light
it was the season of darkness
it was the spring of hope
it was the winter of despair
    ⋮
```

Keyword-in-context search

Given a text of n characters, preprocess it to enable fast substring search (find all occurrences of query string context).

```
% java KWIC tale.txt 15
search
o st giless to search for contraband
her unavailing search for your fathe
le and gone in search of her husband
t provinces in search of impoverishe
 dispersing in search of other carri
n that bed and search the straw hold

better thing
t is a far far better thing that i do than
 some sense of better things else forgotte
was capable of better things mr carton ent
```

number of characters of surrounding context

Applications. Linguistics, databases, web search, word processing, ...

Suffix sort

input string

i	t	w	a	s	b	e	s	t	i	t	w	a	s	w
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

form suffixes

0	i	t	w	a	s	b	e	s	t	i	t	w	a	s	w
1	t	w	a	s	b	e	s	t	i	t	w	a	s	w	
2	w	a	s	b	e	s	t	i	t	w	a	s	w		
3	a	s	b	e	s	t	i	t	w	a	s	w			
4	s	b	e	s	t	i	t	w	a	s	w				
5	b	e	s	t	i	t	w	a	s	w					
6	e	s	t	i	t	w	a	s	w						
7	s	t	i	t	w	a	s	w							
8	t	i	t	w	a	s	w								
9	i	t	w	a	s	w									
10	t	w	a	s	w										
11	w	a	s	w											
12	a	s	w												
13	s	w													
14	w														

sort suffixes to bring query strings together

3	a	s	b	e	s	t	i	t	w	a	s	w			
12	a	s	w												
5	b	e	s	t	i	t	w	a	s	w					
6	e	s	t	i	t	w	a	s	w						
0	i	t	w	a	s	b	e	s	t	i	t	w	a	s	w
9	i	t	w	a	s	w									
4	s	b	e	s	t	i	t	w	a	s	w				
7	s	t	i	t	w	a	s	w							
13	s	w													
8	t	i	t	w	a	s	w								
1	t	w	a	s	b	e	s	t	i	t	w	a	s	w	
10	t	w	a	s	w										
14	w														
2	w	a	s	b	e	s	t	i	t	w	a	s	w		
11	w	a	s	w											

array of suffix indices
in sorted order

Keyword-in-context search: suffix-sorting solution

- Preprocess: suffix sort the text.
- Query: binary search for query; scan until mismatch.

KWIC search for "search" in Tale of Two Cities

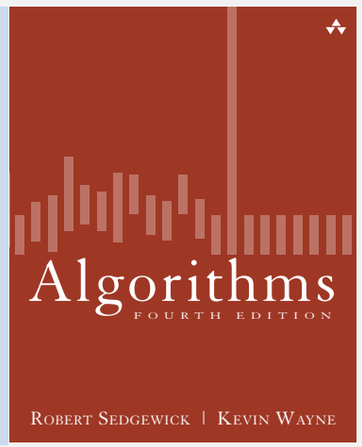
```
                                ⋮
632698  s e a l e d _ m y _ l e t t e r _ a n d _ …
713727  s e a m s t r e s s _ i s _ l i f t e d _ …
660598  s e a m s t r e s s _ o f _ t w e n t y _ …
 67610  s e a m s t r e s s _ w h o _ w a s _ w i …
→ 4430  s e a r c h _ f o r _ c o n t r a b a n d …
 42705  s e a r c h _ f o r _ y o u r _ f a t h e …
499797  s e a r c h _ o f _ h e r _ h u s b a n d …
182045  s e a r c h _ o f _ i m p o v e r i s h e …
143399  s e a r c h _ o f _ o t h e r _ c a r r i …
411801  s e a r c h _ t h e _ s t r a w _ h o l d …
158410  s e a r e d _ m a r k i n g _ a b o u t _ …
691536  s e a s _ a n d _ m a d a m e _ d e f a r …
536569  s e a s e _ a _ t e r r i b l e _ p a s s …
484763  s e a s e _ t h a t _ h a d _ b r o u g h …
                                ⋮
```

War story

Q. How to efficiently form (and sort) the n suffixes?

```
String[] suffixes = new String[n];
for (int i = 0; i < n; i++)
    suffixes[i] = s.substring(i, n);

Arrays.sort(suffixes);
```



3rd printing (2012)

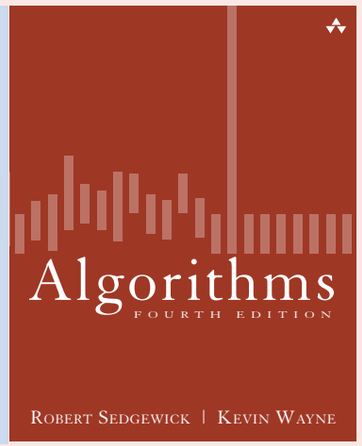
input file	characters	Java 7u5	Java 7u6
amendments.txt	18 K	0.25 sec	2.0 sec
aesop.txt	192 K	1.0 sec	*out of memory*
mobydick.txt	1.2 M	7.6 sec	*out of memory*
chromosome11.txt	7.1 M	61 sec	*out of memory*

Radix sorting: quiz 3

How much memory as a function of n?

```
String[] suffixes = new String[n];
for (int i = 0; i < n; i++)
    suffixes[i] = s.substring(i, n);

Arrays.sort(suffixes);
```



3rd printing (2012)

A. 1

B. n

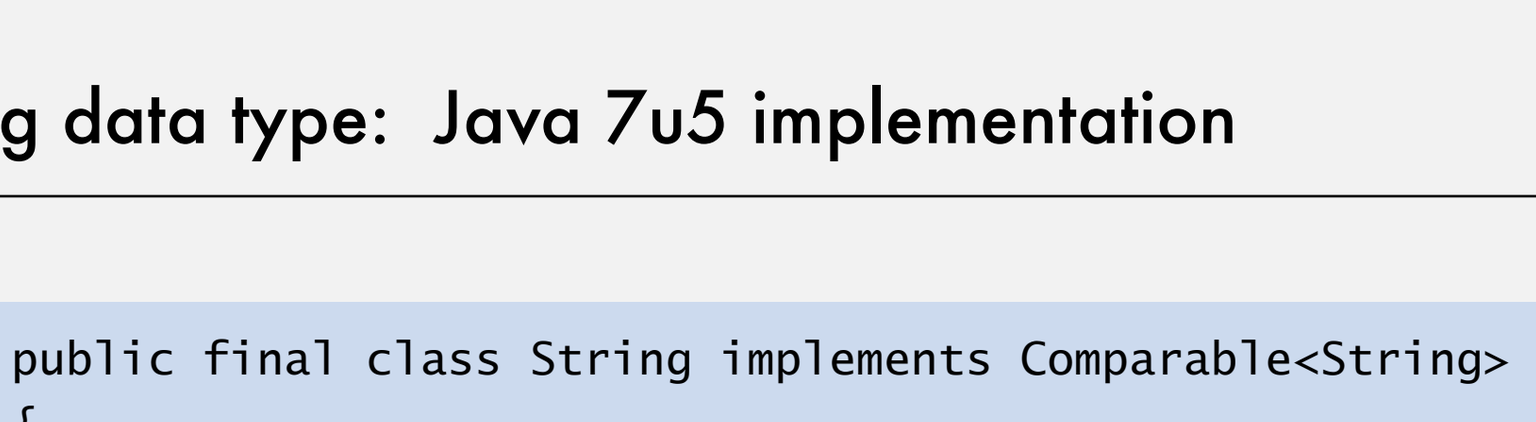
C. $n \log n$

D. n^2

The String data type: Java 7u5 implementation

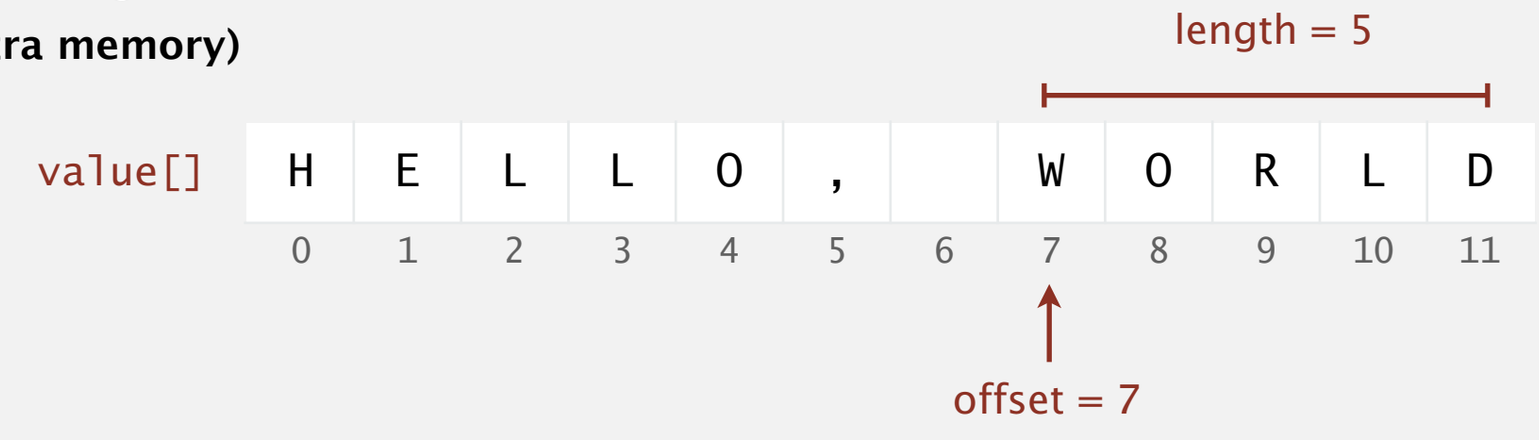
```
public final class String implements Comparable<String>
{
   private char[] value;  // characters
   private int offset;    // index of first char in array
   private int length;    // length of string
   private int hash;      // cache of hashCode()
   …
```

String s = "Hello, World";



String t = s.substring(7, 12);

(constant extra memory)



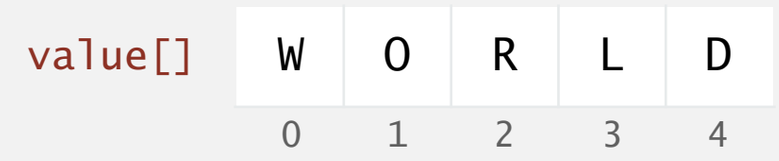
The String data type: Java 7u6 implementation

```
public final class String implements Comparable<String>
{
   private char[] value;  // characters
   private int hash;      // cache of hashCode()
   …
```

String s = "Hello, World";

String t = s.substring(7, 12);

(linear extra memory)



The String data type: performance

String data type (in Java). Sequence of characters (immutable).

Java 7u5. Immutable `char[]` array, offset, length, hash cache.

Java 7u6. Immutable `char[]` array, hash cache.

operation	Java 7u5	Java 7u6
length	1	1
indexing	1	1
concatenation	$m + n$	$m + n$
substring extraction	1	n
immutable?	✔	✔
memory	$64 + 2n$	$56 + 2n$

A Reddit exchange

I'm the author of the substring() change. As has been suggested in the analysis here there were two motivations for the change

- Reduce the size of String instances. Strings are typically 20-40% of common apps footprint.
- Avoid memory leakage caused by retained substrings holding the entire character array.

bondolo

Changing this function, in a bugfix release no less, was totally irresponsible. It broke backwards compatibility for numerous applications with errors that didn't even produce a message, just freezing and timeouts... All pain, no gain. Your work was not just vain, it was thoroughly destructive, even beyond its immediate effect.

cypherpunks

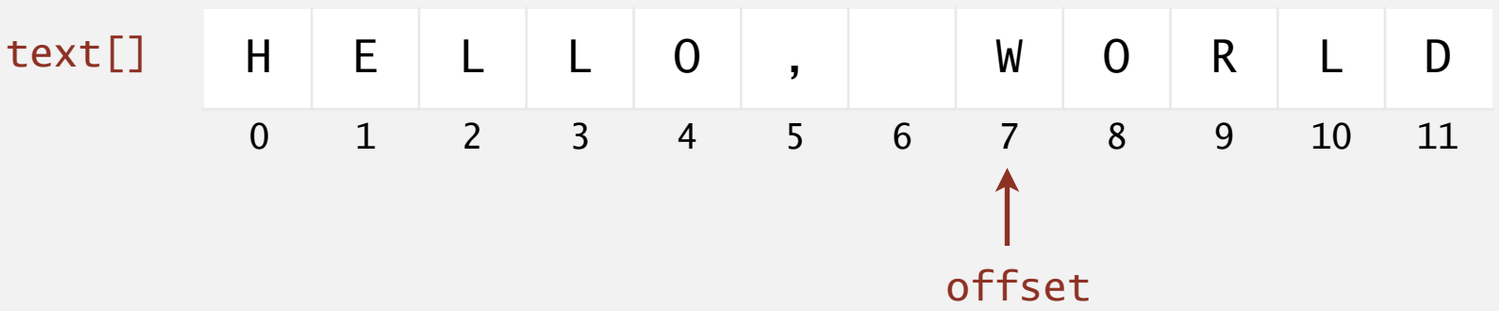
http://www.reddit.com/r/programming/comments/1qw73v/til_oracle_changed_the_internal_string

Suffix sort

Q. How to efficiently form (and sort) suffixes in Java 7u6?

A. Define `Suffix` class ala Java 7u5 `String` representation.

```
public class Suffix implements Comparable<Suffix>
{
   private final String text;
   private final int offset;
   public Suffix(String text, int offset)
   {
      this.text = text;
      this.offset = offset;
   }
   public int length()               { return text.length() - offset;   }
   public char charAt(int i)         { return text.charAt(offset + i);  }
   public int compareTo(Suffix that) { /* see textbook */                }
}
```



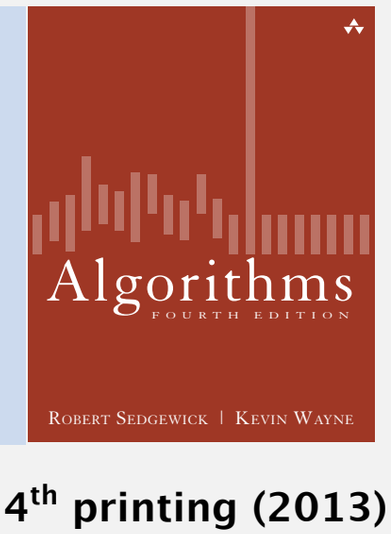
Suffix sort

Q. How to efficiently form (and sort) suffixes in Java 7u6?

A. Define `Suffix` class ala Java 7u5 `String` representation.

```
Suffix[] suffixes = new Suffix[n];
for (int i = 0; i < n; i++)
    suffixes[i] = new Suffix(s, i);

Arrays.sort(suffixes);
```



4th printing (2013)

Optimizations. [5× faster and 32× less memory than Java 7u5 version]

- Use 3-way string quicksort instead of `Arrays.sort()`.
- Manipulate suffix offsets directly instead of via explicit `Suffix` objects.

Suffix arrays: theory

Conjecture. [Knuth 1970] No linear-time algorithm.

Proposition. [Weiner 1973] Linear-time algorithms (suffix trees).

" has no practical virtue... but a historic monument in the area of string processing. "

LINEAR PATTERN MATCHING ALGORITHMS

Peter Weiner

The Rand Corporation, Santa Monica, California*

Abstract

In 1970, Knuth, Pratt, and Morris [1] showed how to do basic pattern matching in linear time. Related problems, such as those discussed in [4], have previously been solved by efficient but sub-optimal algorithms. In this paper, we introduce an interesting data structure called a bi-tree. A linear time algorithm for obtaining a compacted version of a bi-tree associated with a given string is presented. With this construction as the basic tool, we indicate how to solve several pattern matching problems, including some from [4], in linear time.

A Space-Economical Suffix Tree Construction Algorithm

EDWARD M. McCREIGHT

Xerox Palo Alto Research Center, Palo Alto, California

ABSTRACT. A new algorithm is presented for constructing auxiliary digital search trees to aid in exact-match substring searching. This algorithm has the same asymptotic running time bound as previously published algorithms, but is more economical in space. Some implementation considerations are discussed, and new work on the modification of these search trees in response to incremental changes in the strings they index (the update problem) is presented.

On–line construction of suffix trees [1]

Esko Ukkonen

Department of Computer Science, University of Helsinki,
P. O. Box 26 (Teollisuuskatu 23), FIN–00014 University of Helsinki, Finland
Tel.: +358-0-7084172, fax: +358-0-7084441
Email: ukkonen@cs.Helsinki.FI

Suffix arrays: practice

Applications. Bioinformatics, information retrieval, data compression, ...

Many ingenious algorithms.

- Constants and memory footprint very important.
- State-of-the art still changing.

year	algorithm	worst case	memory	
1991	Manber–Myers	$n \log n$	$8\,n$	← see lecture videos
1999	Larsson–Sadakane	$n \log n$	$8\,n$	← about 10× faster than Manber–Myers
2003	Kärkkäinen–Sanders	n	$13\,n$	
2003	Ko–Aluru	n	$10\,n$	
2008	divsufsort2	$n \log n$	$5\,n$	← good choices (libdivsufsort)
2010	sais	n	$6\,n$	