Algorithms

Robert Sedgewick | Kevin Wayne

http://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*

Algorithms

Robert Sedgewick | Kevin Wayne

http://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*

String processing

String. Sequence of characters.

Important fundamental abstraction.

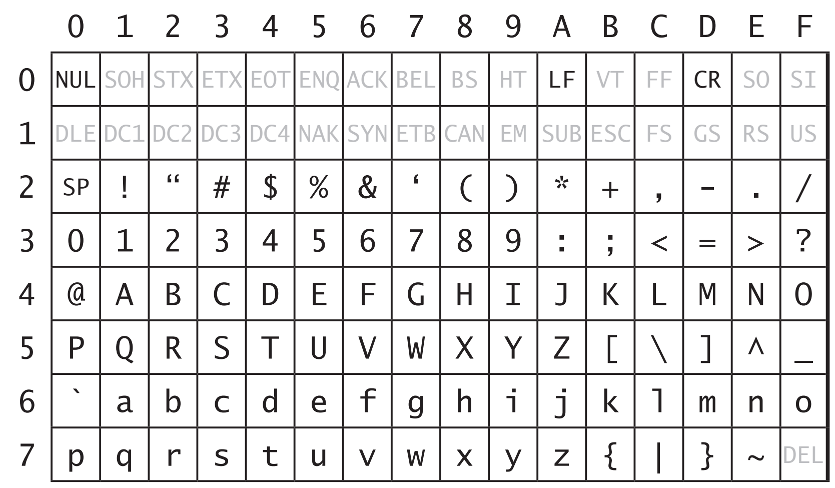
- Information processing.
- Genomic sequences.
- Communication systems (e.g., email).
- Programming systems (e.g., Java programs).
- …

> *“ The digital information that underlies biochemistry, cell biology, and development can be represented by a simple string of G's, A's, T's and C's. This string is the root data structure of an organism's biology. ”* — *M. V. Olson*

The char data type

C char data type. Typically an 8-bit integer.

- Supports 7-bit ASCII.
- Can represent only 256 characters.



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SP	!	“	#	$	%	&	‘	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Hexadecimal to ASCII conversion table

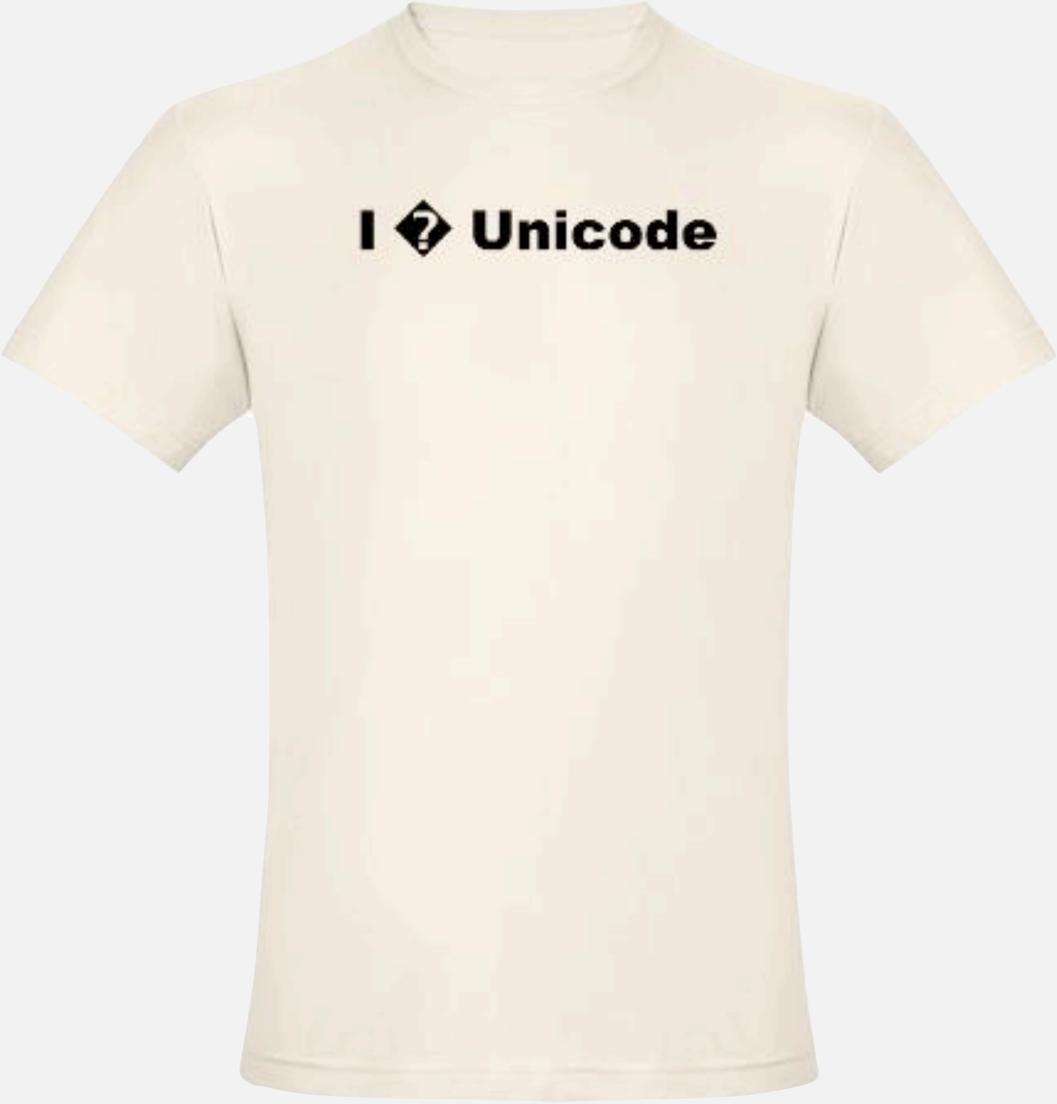
A	á	∂	𝔊
U+0041	U+00E1	U+2202	U+1D50A

Unicode characters

Java char data type. A 16-bit unsigned integer.

- Supports original 16-bit Unicode.
- Supports 21-bit Unicode 3.0 (awkwardly).

I (heart) Unicode



The String data type

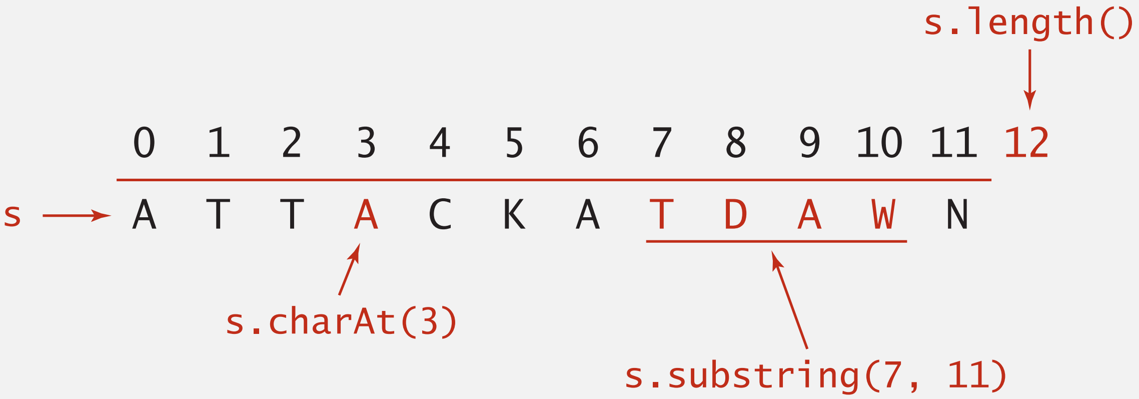
String data type in Java. Sequence of characters (immutable).

Length. Number of characters.

Indexing. Get the i^{th} character.

Substring extraction. Get a contiguous subsequence of characters.

String concatenation. Append one character to end of another string.



The String data type: Java implementation

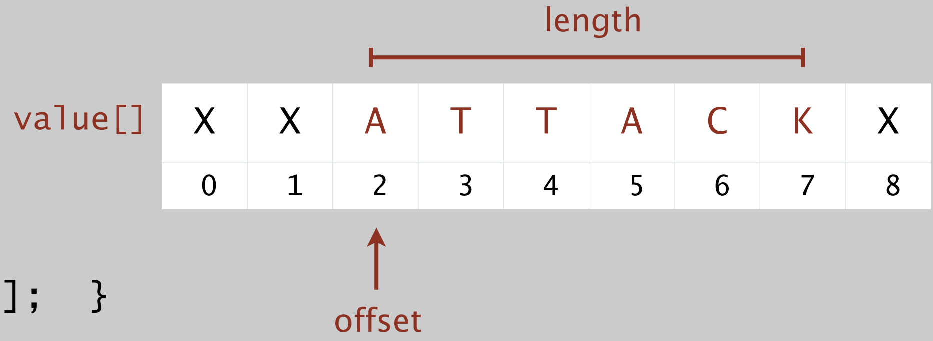
```
public final class String implements Comparable<String>
{
   private char[] value;  // characters
   private int offset;    // index of first char in array
   private int length;    // length of string
   private int hash;      // cache of hashCode()

   public int length()
   {  return length; }

   public char charAt(int i)
   {  return value[i + offset];  }


   private String(int offset, int length, char[] value)
   {
      this.offset = offset;
      this.length = length;
      this.value  = value;
   }

   public String substring(int from, int to)
   {  return new String(offset + from, to - from, value);  }
   …
```



X	X	A	T	T	A	C	K	X
0	1	2	3	4	5	6	7	8

copy of reference to original char array

The String data type: performance

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

Underlying implementation. Immutable `char[]` array, offset, and length.

	String	
operation	guarantee	extra space
`length()`	1	1
`charAt()`	1	1
`substring()`	1	1
`concat()`	N	N

Memory. $40 + 2N$ bytes for a virgin `String` of length N.

can use `byte[]` or `char[]` instead of `String` to save space
(but lose convenience of `String` data type)

The StringBuilder data type

StringBuilder data type. Sequence of characters (mutable).

Underlying implementation. Resizing `char[]` array and length.

	String		StringBuilder	
operation	guarantee	extra space	guarantee	extra space
`length()`	1	1	1	1
`charAt()`	1	1	1	1
`substring()`	1	1	N	N
`concat()`	N	N	1 *	1 *

* amortized

Remark. `StringBuffer` data type is similar, but thread safe (and slower).

String vs. StringBuilder

Q. How to efficiently reverse a string?

A.

```
public static String reverse(String s)
{
   String rev = "";
   for (int i = s.length() - 1; i >= 0; i--)
      rev += s.charAt(i);
   return rev;
}
```

quadratic time

B.

```
public static String reverse(String s)
{
   StringBuilder rev = new StringBuilder();
   for (int i = s.length() - 1; i >= 0; i--)
      rev.append(s.charAt(i));
   return rev.toString();
}
```

linear time

String challenge: array of suffixes

Q. How to efficiently form array of suffixes?

input string

a	a	c	a	a	g	t	t	t	a	c	a	a	g	c
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

suffixes

0	a	a	c	a	a	g	t	t	t	a	c	a	a	g	c
1	a	c	a	a	g	t	t	t	a	c	a	a	g	c	
2	c	a	a	g	t	t	t	a	c	a	a	g	c		
3	a	a	g	t	t	t	a	c	a	a	g	c			
4	a	g	t	t	t	a	c	a	a	g	c				
5	g	t	t	t	a	c	a	a	g	c					
6	t	t	t	a	c	a	a	g	c						
7	t	t	a	c	a	a	g	c							
8	t	a	c	a	a	g	c								
9	a	c	a	a	g	c									
10	c	a	a	g	c										
11	a	a	g	c											
12	a	g	c												
13	g	c													
14	c														

String vs. StringBuilder

Q. How to efficiently form array of suffixes?

A.

```
public static String[] suffixes(String s)
{
   int N = s.length();
   String[] suffixes = new String[N];
   for (int i = 0; i < N; i++)
      suffixes[i] = s.substring(i, N);
   return suffixes;
}
```

linear time and
linear space

B.

```
 public static String[] suffixes(String s)
 {
    int N = s.length();
    StringBuilder sb = new StringBuilder(s);
    String[] suffixes = new String[N];
    for (int i = 0; i < N; i++)
       suffixes[i] = sb.substring(i, N);
    return suffixes;
}
```

quadratic time and
quadratic space

Longest common prefix

Q. How long to compute length of longest common prefix?

p	r	e	f	e	t	c	h
0	1	2	3	4	5	6	7
p	r	e	f	i	x		

```
public static int lcp(String s, String t)
{
   int N = Math.min(s.length(), t.length());
   for (int i = 0; i < N; i++)
      if (s.charAt(i) != t.charAt(i))
         return i;
   return N;
}
```

linear time (worst case)
sublinear time (typical case)

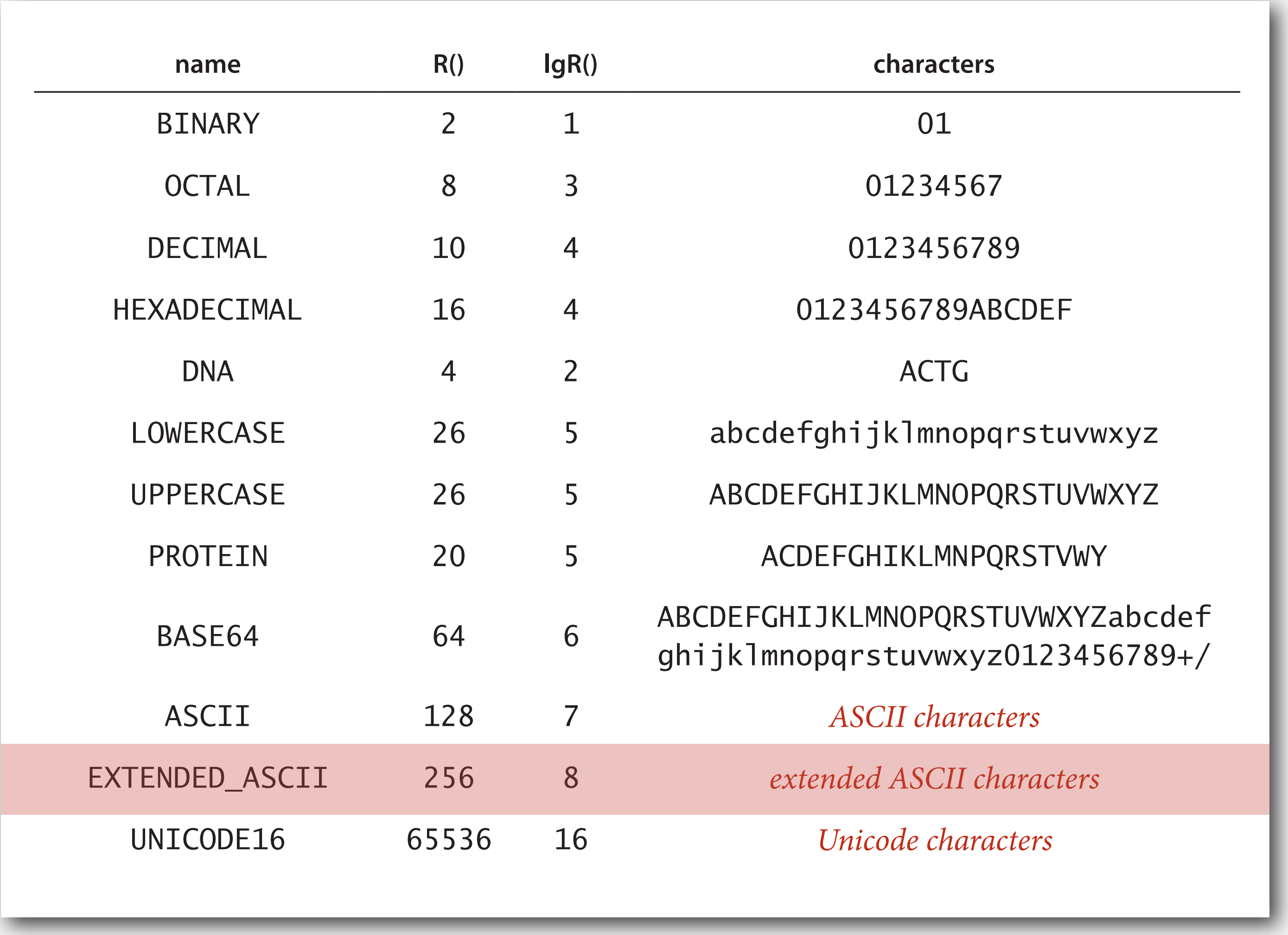
Running time. Proportional to length D of longest common prefix.

Remark. Also can compute `compareTo()` in sublinear time.

Alphabets

Digital key. Sequence of digits over fixed alphabet.

Radix. Number of digits R in alphabet.



name	R()	lgR()	characters
BINARY	2	1	01
OCTAL	8	3	01234567
DECIMAL	10	4	0123456789
HEXADECIMAL	16	4	0123456789ABCDEF
DNA	4	2	ACTG
LOWERCASE	26	5	abcdefghijklmnopqrstuvwxyz
UPPERCASE	26	5	ABCDEFGHIJKLMNOPQRSTUVWXYZ
PROTEIN	20	5	ACDEFGHIKLMNPQRSTVWY
BASE64	64	6	ABCDEFGHIJKLMNOPQRSTUVWXYZabcdef ghijklmnopqrstuvwxyz0123456789+/
ASCII	128	7	*ASCII characters*
EXTENDED_ASCII	256	8	*extended ASCII characters*
UNICODE16	65536	16	*Unicode characters*