Chapter 8, Strings:

- General introduction; string sorting (Today)
- Tries (Friday)
- Other string topics (next week Monday and Wednesday)
  - Regular expressions
  - Huffman encoding
  - Edit distance
  - Grammars and parsing

Today:

- End-of-semester business
- Sorting strings
  - Why we care about strings
  - String quick sort
  - String bucket sort
  - String radix sort
Projects:

- Last regular project score update on Tues, Dec 6
- “Two minute warning” run of scripts on Fri, Dec 9 (no Schoology update—see report file)
- All projects due on the last day of classes, midnight between Fri, Dec 9 and Sat, Dec 10—not last day of finals.

Final exam

- Our final exam block is Wed, Dec 14, 10:30am–12:30pm
- During our final exam block, we will meet in the CSCI lab
- Test 3 ("written"/conceptual part) will be like Test 1, but covering BSTs (ch 5) through strings (ch 8)
- Test 4 (programming part) will work the same way as Test 2, covering dynamic programming, hashing, and strings.
Why we care about strings

- Strings are different
- Strings are common
- Strings are a representative example

```java
public class DNASequence {
    /** An alphabet for DNA */
    private static enum Nucleotide { A, C, G, T }
    /** The string of nucleotides */
    private Nucleotide[] sequence;
}
```
public class BigInt {

    private byte[] digits;

    /** Compute the sum of this and another BigInt. */
    public BigInt add(BigInt other) {
        // The result object
        BigInt sum = new BigInt();
        // The result object has at most one more digit
        // than the larger number of digits of the two addends
        sum.digits = new byte[(digits.length > other.digits.length?
            digits.length : other.digits.length) + 1];
        // Add by column
        int carry = 0;
        for (int i = 0; i < sum.digits.length; i++) {
            // Digits in current columns of the two addends
            int a = digits.length <= i? digits[i] : 0;
            int b = other.digits.length <= i ? other.digits.length : 0;
            // The sum of the current digits plus carry from previous iteration
            int s = a + b + carry;
            // Mod that sum by 256 to get the appropriate digit in result,
            // divide to get the carry for next time.
            sum.digits[i] = (byte) (s % 256);
            carry = s / 256;
        }

        assert carry == 0;

        return sum;
    }
}
struct employee
{
    char surname[20];
    char first_name[20];
    double salary;
    char extension[4]
};

struct book
{
    char title[100];
    char author[50];
    int pages;
    char call_number[8];
    int status;
};

struct complex_number { double real, double imag };
Quick sort:

\[ i, j \]

\[
\begin{array}{cccccccccccc}
\cdots & 91 & 88 & 44 & 62 & 56 & 33 & 59 & 31 & 59 & 53 & \cdots \\
start & & & & & & & & & & stop & \\
\end{array}
\]

unsearched

Invariant 11 (Loop of partition())

(a) \( \text{start} \leq i \leq j < \text{stop} \).
(b) \( \forall k \in [\text{start}, i), \text{sequence}[k] < \text{sequence}[\text{stop} - 1] \).
(c) \( \forall k \in [i, j), \text{sequence}[k] \geq \text{sequence}[\text{stop} - 1] \).
(d) \( j - \text{start} \) is the number of iterations completed.
<table>
<thead>
<tr>
<th>dais</th>
<th>card</th>
<th>bark</th>
<th>care</th>
<th>even</th>
<th>barb</th>
<th>doze</th>
<th>cart</th>
<th>carb</th>
<th>axle</th>
<th>daze</th>
<th>exam</th>
<th>axis</th>
<th>bard</th>
<th>carp</th>
</tr>
</thead>
<tbody>
<tr>
<td>card</td>
<td>bark</td>
<td>care</td>
<td>barb</td>
<td>carb</td>
<td>axle</td>
<td>axis</td>
<td>bard</td>
<td>carp</td>
<td>dais</td>
<td>even</td>
<td>doze</td>
<td>cart</td>
<td>daze</td>
<td>exam</td>
</tr>
<tr>
<td>barb</td>
<td>axle</td>
<td>axis</td>
<td>bard</td>
<td>card</td>
<td>bark</td>
<td>care</td>
<td>carb</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
i \quad j \quad k
\]

<table>
<thead>
<tr>
<th>bark</th>
<th>barb</th>
<th>card</th>
<th>care</th>
<th>cart</th>
<th>dais</th>
<th>even</th>
<th>doze</th>
<th>carb</th>
<th>axle</th>
<th>daze</th>
<th>exam</th>
<th>axis</th>
<th>bard</th>
<th>carp</th>
</tr>
</thead>
</table>

| <pivot | =pivot | ≥pivot | unsearched |
Invariant 37. [Loop of string_quick_sort_r()]
Let $c$ be the character in position $\text{pre}$ in the string in position $\text{stop} - 1$.

(a) $\text{start} \leq i \leq j \leq k < \text{stop}$

(b) (Informal) For all the strings in range $[\text{start}, i)$, their character in position $\text{pre}$ is less than $c$.

(c) (Informal) For all the strings in range $[i, j)$, their character in position $\text{pre}$ is equal to $c$.

(d) (Informal) For all the strings in range $[i, j)$, their character in position $\text{pre}$ is greater than to $c$.

(e) $k - \text{start}$ is the number of iterations completed.
Invariant 38. [Precondition of `string_quick_sort_r()`]
∀ i, j ∈ [start, stop), ∀k ∈ [0, pre), sequence[i][k] = sequence[j][k].
beach event can core hope any front ball done a frond an i give eve

can core hope any ball done a an i give eve frond beach event front

can any a an i eve beach core hope done give ball frond event front

a an i beach eve event ball can done frond front hope core give any

a i ball can beach give an any done hope core frond front eve event

a an any ball beach can core done eve event frond front give hope i
Coming up:

*Do Open Addressing project (suggested by Friday, Dec 2)*

*Due Wed, Nov 30 (end of day)*
*Read Section 8.1*
*Do Exercises 8.(4 & 5)*

*Due Thurs, Dec 1*
*Take quiz (on Section 8.1)*

*Due Fri, Dec 2*
*Do Exercises 8.(7, 14, 20)*
*Read Section 8.2*