binary search, linear search

Do exercises on pp. 311-312 to get familiar with concepts and develop skill. Practice in DrJava! Test your methods!

Time spent on A5:

<table>
<thead>
<tr>
<th>Time</th>
<th>Min</th>
<th>Average</th>
<th>Median</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>00-02</td>
<td>04</td>
<td>7.14</td>
<td>6</td>
<td>15</td>
</tr>
<tr>
<td>03-04</td>
<td>57</td>
<td>11-12: 08</td>
<td>13-14: 05</td>
<td>15-16: 04</td>
</tr>
<tr>
<td>04-05</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>05-06</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>06-07</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>07-08</td>
<td>28</td>
<td>11-12: 08</td>
<td>13-14: 05</td>
<td>15-16: 04</td>
</tr>
<tr>
<td>08-09</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>09-10</td>
<td>15</td>
<td>11-12: 08</td>
<td>13-14: 05</td>
<td>15-16: 04</td>
</tr>
<tr>
<td>10-11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11-12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12-13</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13-14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14-15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15-16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16-17</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17-18</td>
<td>06</td>
<td>11-12: 08</td>
<td>13-14: 05</td>
<td>15-16: 04</td>
</tr>
</tbody>
</table>

Comments on A5

Recursion: Make requirements/descriptions less ambiguous, clearer; give more direction. Need optional problem with more complicated recursive solution would have been an interesting challenge, more recursive functions. They make us think. Make task 5 easier. I could not finish it.

Add more methods; it did not take long. Allow us to do recursive methods with loops rather than recursively.

I had intended here to erupt in largely incoherent rage over that wretched concept of recursion, which I came to hate like an enemy: like a sentient being who, knowing the difference between right and wrong, had purposely chosen to do me harm. However, I then figured out how it works, and it is actually quite elegant, so now I suppose I have learned something against my will.

Sorting:

```java
/** Sort b[h..k] */
public static void qsort(int[] b, int h, int k) {
    if (b[h..k] has fewer than 2 elements) return;
    int j = partition(b, h, k);
    // b[h..j-1] <= b[j] <= b[j+1..k]
    // Sort b[h..j-1] and b[j+1..k]
    qsort(b, h, j-1); qsort(b, j+1, k);
}
```

Quicksort

Thought of Quicksort in ~1958. Tried to explain it to a colleague, but couldn’t. Few months later: he saw a draft of the definition of the language Algol 58—later turned into Algol 60. It had recursion. He went and explained Quicksort to his colleague, using recursion, who now understood it.

Tony Hoare, in 1968
Quicksort author

Tony Hoare, in 2007
in Germany
I don't like how we are forced to visualize things in Dr. Gries' way. … Entire point of programming is to be able to look at things in different ways and come up with different solutions for one problem. Forcing us to think of things in his way and testing us on it has been detrimental to my learning because in my opinion it wastes time and confuses me. This course should focus more on solving problems rather than drawing folders to represent objects.

1. A model of execution of Java programs is needed in order to bring understanding.
2. Problem solving is the focus. The programs you wrote for A5, the algorithms we are now studying, and the way we develop them, could not have been possible without the basics that we have given you.
3. We are giving you tools for coming up with good solutions, not just different ones.

1. A model of execution of Java programs is needed in order to bring understanding.
2. Problem solving is the focus. The programs you wrote for A5, the algorithms we are now studying, and the way we develop them, could not have been possible without the basics that we have given you.
3. We are giving you tools for coming up with good solutions, not just different ones.

Software crisis:
Academic and industrial people. Admitted for first time that they did not know how to develop software efficiently and effectively.

Next 10-15 years: intense period of research of software engineering, language design, proving programs correct, etc.

During 1970s, 1980s, intense research on
How to prove programs correct,
How to make it practical,
Methodology for developing algorithms

The way we understand recursive methods is based on that methodology.
Our understanding of and development of loops is based on that methodology.

Throughout, we try to give you thought habits to help you solve programming problems for effectively

Mark Twain: Nothing needs changing so much as the habits of others.

Simplicity is key:
Learn not only to simplify, learn not to complify.

Separate concerns, and focus on one at a time.

Develop and test incrementally.

Don’t solve a problem until you know what the problem is (give precise and thorough specs).

Learn to read a program at different levels of abstraction.