Foundations of computing: Algorithms and Data Structures

Rasmus Pagh

August 29, 2011

Each week we pose exercises related to the lecture of that week. Some exercises will be selected from the textbook (see the course schedule), and others supplied on PDF exercise sheets. Usually, it will be best to *start off with some exercises from the book*, which tend to be less challenging (but not always easy!). Note that some of today's exercises assume that you have set up a working (Java) programming environment, see below.

You will work on the exercises during the exercise session 14-16, and on your own before next week's exercises. Selected exercises from the previous week will be discussed at the beginning of each exercise session. Please provide us with feedback on which exercises you would most like to see no later than Sunday night. Of course, you are always welcome to ask questions about exercises, even if they are not selected.

1 Setting up a programming environment

You should set up your environment for compiling and running Java programs that use the standard input and output libraries that comes with the book. If you are unfamiliar with changing up environment variables in your operating system, you may want to do this during the exercises in case there are problems. Jesper Larsson made most of the following instructions:

1. Download `stdlib.jar` from `http://algs4.cs.princeton.edu/` (under Web Resources, Code).

2. To compile and run Java files from the command prompt you must have the Java Development Kit (JDK) properly installed on the computer. If you have Eclipse installed, the files you need should be somewhere inside that installation, but it may be easier to download another copy of the JDK from `java.com`. If you cannot run `javac` from the command prompt after you have installed the JDK, you need to set the execution path to include the `bin` directory of the JDK, which is where you find the `javac` command. On a Mac this seems to happen automatically, but on Windows you may have to do it manually. One way is to issue a command like this at the command prompt:

   ```
   set PATH=%PATH%;C:\Program Files\java\jdk1.6.0_xx\bin.
   ```

   There should be no quotation marks, and no spaces around the equals sign. The exact path to the bin directory may differ, depending on where the installation put the files, and what version of the JDK you are using. However, the mainstream way to set the PATH in newer versions of Windows is to go via the control panel and edit the value of the PATH you find there. This will set the path permanently. If you do this, be careful not to delete the previous contents! You should just add something to the end like:

   ```
   ;C:\Program Files\java\jdk1.6.0_xx\bin.
   ```

   Again, this is not exactly what you should add. You have to check the directory names on your computer.

3. Set the `CLASSPATH` to include the both the current directory (symbolized by a dot) and the file `stdlib.jar`. Again on Windows you can either edit (or create, if none exists) this value via the control panel, or you can issue a command like this:

   ```
   set CLASSPATH=.;c:\Wherever I Put The File stdlib.jar\.
   ```

   On a Mac you can place `stdlib.jar` in `/Library/Java/Extensions/`, which is in the classpath. In Linux, the command looks something like this:

   ```
   export CLASSPATH='.:/Wherever I Put The File/tdlib.jar'
   ```

   The single quotes are only needed if you used some special characters in the path; in this case the spaces. And if you do use them, it must be the ordinary straight quotes that you find on your keyboard, not some fancy-typography version.

2 Additional runtime analysis exercises

Analyze the asymptotic (tilde notation) best-case and worst-case running times of the pseudocode fragments below (function of n). An answer that the algorithm has the same running time as "well-known algorithm X" is acceptable.

1.
```
1: for i = 1 to n do
2:     for j = 1 to n do
3:         for k = 1 to n do
4:             if (i < j) and (j < k) and A[i] < A[j] and A[j] < A[k] then  c = c + 1;
```

2.
```
1: for i := 1 to ⌊n/2⌋ do
2:     for j := i + 1 to n − i do
3:         if A[i]+A[j]==A[i+j] then
4:             return true
5:         end if
6:     end for
7: end for
8: return false
```

3.
```
1: for i:=1 to n do
2:     if i ∗ i ≥ n then
3:         return i
4:     end if
5: end for
```

4.
```
1: l := 0
2: r := n
3: while l < r − 1 do
4:     m := ⌊(l + r)/2⌋
5:     if A[l]*A[m] < 0 then
6:         r:=m
7:     else
8:         l:=m
9:     end if
10: end while
11: return m
```