Design and Analysis of Algorithms

2017/2018

Independent Project

This assignment consists of:

i. An implementation, from scratch, of a non-trivial algorithm;
ii. A detailed mathematical analysis of its correctness, space and time usage;
iii. The identification and explanation of inputs that result in worst-, average- and best-case performance;
iv. A numeric characterization of the algorithm worst- and average-case space and time usage.

These results should be presented in a (max) 10-page report, with figures and citations. The grade will be determined by the quality of the analysis, experimental results, correctness, quality, and tuning of the source code, and overall written presentation. The presence of any code that is not original to the submitting student will result in a failing grade. All submitted code is subject to Googling.

In numerically characterizing the space and time performance *(iv)*, you must implement an appropriate randomized input generator, describe it in your report, and use it to demonstrate that your implementation achieves the claimed asymptotic bounds on both space and time presented in *(ii)* across input sizes that vary over several orders of magnitude. You may inspire yourself in the *Observations* section of *Sedgewick & Wayne* slides, contained in the `14AnalysisOfAlgorithms.pdf` file. To measure space usage, you can use the `RunningTime` class in Java library (an example at the end of this document).

The report should clearly explain the type of problems the algorithm solves, the idea behind the algorithm, analytic results, pseudo-code, and it should both describe the setup and comment on the results of the numerical tests. It should further include an explanation on how to run the algorithm on a given input, and how to run the infrastructure for numerical evaluation. Do not forget to run each experiment a few times and take the average of the system times.

Your source code should be appropriately commented (*javadoc*) so that I can understand what you are doing and why, and it must be runable – that is, if I try to compile and run it, it should work as advertised.

A *zip* file containing the report in *pdf* format and the various classes containing the source code must be uploaded through *Moodle* by 23:55, May 7th. Your *zip* file should be named `DAAxxxxx.zip`, where `xxxxx` is your student number.

The algorithm

The algorithm is "Shipping n supplies by air with minimum cost" and should solve the following problem:

Suppose you are Consulting for a company that manufactures PC equipment and ships it to distributors all over the country. For each of the next n weeks, they have a projected *supply* s_i of equipment (measured in kilograms), which has to be shipped by an air freight carrier.

Each week's supply can be carried by one of two air freight companies, A or B.

- Company A charges a fixed rate r per kilogram (so it costs $r.s_i$ to ship a week's supply s_i).
- Company B makes contracts for a fixed amount c per week, independent of the weight. However, contracts with company B must be made in blocks of four consecutive weeks at a time.

A *schedule*, for the PC company, is a choice of air freight company (*A* or *B*) for each of the *n* weeks, with the restriction that company *B*, whenever it is chosen, must be chosen for blocks of four contiguous weeks at a time. The *cost of the schedule* is the total amount paid to companies *A* and *B*, according to the description above.

Example: suppose $r = 1$, $c = 10$ and the sequence of projected supplies s_i is *11,9,9,12,12,12,12,9,9,11.* Then the optimal schedule would be to choose company *A* for the first three weeks, then company *B* for a block of four consecutive weeks, and the company *A* for the final three weeks *(AAABBBBAAA)*.

Given a sequence of supply values s_1, s_2, ... s_n as input, the algorithm must return a schedule of minimum cost in polynomial time.

Example on measuring a program space usage (in *StackOverflow.com*):

```
import java.util.ArrayList;
import java.util.List;

public class PerformanceTest {
  private static final long MEGABYTE = 1024L * 1024L;

  public static long bytesToMegabytes(long bytes) {
    return bytes / MEGABYTE;
  }

  public static void main(String[] args) {
    // I assume you will know how to create a object Person yourself...
    List<Person> list = new ArrayList<Person>();
    for (int i = 0; i <= 100000; i++) {
      list.add(new Person("Jim", "Knopf"));
    }
    // Get the Java runtime
    Runtime runtime = Runtime.getRuntime();
    // Run the garbage collector
    runtime.gc();
    // Calculate the used memory
    long memory = runtime.totalMemory() - runtime.freeMemory();
    System.out.println("Used memory is bytes: " + memory);
    System.out.println("Used memory is megabytes: "
                                       + bytesToMegabytes(memory));
  }
}
```