COMP4500/7500
Advanced Algorithms and Data Structures

School of Information Technology and Electrical Engineering
The University of Queensland, Semester 2, 2020

Assignment 2

Due at 4:00pm, Monday the 26th October 2020.
This assignment is worth 20% of your final grade.

This assignment is to be attempted **individually**. Please read this entire handout before attempting any of the questions.

Submission. Answers to each of the questions Q1(b) and Q1(d) should be clearly labelled and included in a pdf file called `a2.pdf`.

You need to submit (i) your written answers in `a2.pdf`, as well as (ii) your source code files `Recursive.java` and `Dynamic.java` electronically using Blackboard according to the exact instructions on the Blackboard website: `https://learn.uq.edu.au/`

You can submit your assignment multiple times before the assignment deadline but only the last submission will be saved by the system and marked. Only submit the files listed above. You are responsible for ensuring that you have submitted the files that you intended to submit in the way that we have requested them. You will be marked on the files that you submitted and not on those that you intended to submit. Only files that are submitted according to the instructions on Blackboard will be marked.

Submitted work should be neat, legible and simple to understand – you may be penalised for work that is untidy or difficult to read and comprehend.

For the programming part, you will be penalised for submitting files that are not compatible with the assignment requirements. In particular, code that is submitted with compilation errors, or is not compatible with the supplied testing framework will receive 0 marks.

Late submission. Unless you have been approved to submit an assignment after the due date: late assignments will lose 10% of the marks allocated to the assignment immediately, and a further 10% of the marks allocated to the assignment for each additional calendar day late. Assignments more than 5 calendar days late will not be accepted.

If there are medical or exceptional circumstances that will affect your ability to complete an assignment by the due date, then you can apply for an extension as per Section 5.3 of the electronic course profile (ECP). Requests must be made at least 48 hours prior to the submission deadline. Assignment extensions longer than 7 calendar days will not be granted.

School Policy on Student Misconduct. You are required to read and understand the School Statement on Misconduct, available at the School's website at:

`http://www.itee.uq.edu.au/itee-student-misconduct-including-plagiarism`

This is an individual assignment. If you are found guilty of misconduct (plagiarism or collusion) then penalties will be applied.

Question 1 (100 marks total)

You have been appointed as a site engineer in a mining company. One of your duties is operating a heavy duty hydraulic slurry pump on site for k consecutive hours. Regular maintenance of the slurry pump helps it function optimally and reliably. It will extend the pump's wear life, helps avoid unplanned downtime, and lowers the cost.

There are several services that can be performed for this pump:

- Minor Service (takes 1 hour). Includes changing oil, inspecting the bearings, looking for leakages and examining the stuffing box.
- Regular Service (takes 2 hours). Includes - in addition to the minor service activities - cleaning and checking belts, as well as changing filters.
- Full Service (takes 4 hours). Involves a full inspection of the entire pump.

During any service, the pump cannot be used to transport any liquid.

The maximum amount (volume) of slurry (liquid) the pump can transport per hour depends on the slurry density. For simplicity, we assume density is constant. The transported volume depends on the last service applied to the pump, and the number of hours since that service.

When the last service was a full service, the maximum volume of liquid that can be transported per hour i hours after that activity has concluded (for $0 \leq i$) is described by an array `fullServiceCapacity` of n non-negative integers (indexed from 0). The hour after the full service activity has finished, the maximum volume per hour of the pump is `fullServiceCapacity[0]`; the i'th hour after the full service activity is finished (i indices start from zero), the volume per hour of the pump is `fullServiceCapacity[`i`]`, where $0 \leq i < n$. n hours after a full service is finished, the pump cannot be operated until another service is performed.

Similar arrays exist for regular and minor services, called `regularServiceCapacity` (length m) and `minorServiceCapacity` (length p) respectively. These describe the maximum volume per hour of the pump i hours after either of these services are completed. Again, if we are m hours after a regular service or p hours after a minor service, then the pump cannot be operated until another service is performed.

You have also been given a schedule, represented by an array `hourlyVolume` of k non-negative integers (indexed from 0). This array gives the volume of liquid that is scheduled to be transported for each of the k hours that you are in charge of the pump. The volume of liquid scheduled to be transported in the first hour (hour 0) is `hourlyVolume[0]`, the volume scheduled for the second hour (hour 1) is `hourlyVolume[1]` etc. If the amount of liquid needing to be transported in an hour (given by `hourlyVolume`) is greater than the amount that can be transported in that hour (zero during any service, otherwise determined by `fullServiceCapacity`, `regularServiceCapacity` or `minorServiceCapacity`), then the remaining liquid must be discarded. Hence **the cost/loss to your company is the amount of liquid that could not be transported**.

A service can be scheduled to take place at the start of any hour you are in charge of the pump. The pump will be out of action for the duration of the service (e.g 2 hours for a regular service) and cannot transport any liquid during this time. Once the service is complete, the pump can begin transporting liquid again. Only one service can be in progress at any given time, but otherwise there is no limit to the number of services that can be performed, or the order in which they can be done. For the k hours that you are in charge, it is up to you to determine when these services should occur.

Given arrays `fullServiceCapacity`, `regularServiceCapacity`, `minorServiceCapacity` and `hourlyVolume`, and the knowledge that the last service performed on the pump was a full service that concluded the hour before you were put in charge of the pump, **your task is to find the least loss that can be incurred by your company over the k hours that you operate the slurry pump.**

Example
As an example, consider the following scenario:

$$\begin{array}{lcl} \texttt{hourlyVolume} & = & [50, 40, 90, 10, 5, 100, 40, 20, 50] \\ \texttt{fullServiceCapacity} & = & [100, 90, 80, 70, 60, 50, 40, 30, 20, 10] \\ \texttt{regularServiceCapacity} & = & [70, 50, 40, 30, 20, 10] \\ \texttt{minorServiceCapacity} & = & [50, 40, 20, 10] \end{array}$$

where you are in charge for $k = 9$ hours, and a full service took place the hour before you were put in charge of the pump.

For each of the k hours (i.e hours $0, 1, 2 \cdots k-1$) you are in charge of the pump, you can either choose to perform one of the three possible services, or no service at all. Your choices can have an impact on the loss incurred by the company. For example,

- If you choose to perform no services at all:
  Hourly Capacity = [100, 90, 80, 70, 60, 50, 40, 30, 20]
  Total loss = 0 + 0 + 10 + 0 + 0 + 50 + 0 + 0 + 30 = 90
- If you choose to perform a full service starting at hour 3 (i.e. the fourth hour):
  Hourly Capacity = [100, 90, 80, 0, 0, 0, 0, 100, 90]
  Total loss = 0 + 0 + 10 + 10 + 5 + 100 + 40 + 0 + 0 = 165
- If you choose to perform a regular service starting at hour 3 (i.e the fourth hour) and a minor service starting at hour 7 (i.e the eighth hour)
  Hourly Capacity = [100, 90, 80, 0, 0, 70, 50, 0, 50]
  Total loss = 0 + 0 + 10 + 10 + 5 + 30 + 0 + 20 + 0 = 75

Where 'hourly capacity' refers to the maximum amount of liquid that could be transported in a given hour. This is zero during any service activity and follows the relevant array after a service is finished.

There are many other possible service schedules. For this example, the least loss that can be incurred by the company for the $k = 9$ hours that you operate the pump is 75.

a. (20 marks) Implement the public static method `optimalLossRecursive` from the `Recursive` class in the `assignment2` package that is available in the zip file that accompanies this handout, to provide a recursive algorithm to determine the least cost that can be incurred by the company over the k hours (i.e. hour 0 to hour $k-1$) that you operate the slurry pump. To implement that method, you will need to provide an implementation for the private static method `optimalLossRecursive` from the same class.

   The recursive solution does NOT need to find a schedule of services that produces the least loss – it just needs to determine the least loss. Efficiency is not at all a concern for this part, so focus on an elegant solution.

b. (20 marks) It is expected that your recursive algorithm will not be polynomial-time in the worst case. For the case where the number of hours that you are responsible for the slurry pump is k, give an asymptotic lower bound on the worst-case time complexity of your recursive algorithm in terms of parameter k. Make your bound as tight as possible.

   Give an argument explaining why the time-complexity is exponential in terms of a (lower bound) recurrence derived from your algorithm.

   [Make your answer as concise as possible – it should be no more than half a page using minimum 11pt font. Longer answers will not be marked.]

c. (30 marks) Develop a bottom-up dynamic programming solution to the problem **(not memoised)** by implementing the public static method `optimalLossDynamic` in the `Dynamic` class from the `assignment2` package that accompanies this handout.

   Your dynamic programming solution should run in polynomial time (in terms of k, the number of hours you are responsible for the slurry pump).

   This dynamic solution does NOT need to find a schedule of activities that produces the least loss – it just needs to determine the least loss.

d. (10 marks) Provide an asymptotic upper bound on the worst-case time complexity of your dynamic programming solution for part (c) in terms of the parameter k, the number of hours you are responsible for the slurry pump. Make your bounds as tight as possible and justify your solution.

   You should assume for this analysis that `hourlyVolume` is the shortest array that you are provided.

   [Make your answer as concise as possible – it should be no more than half a page using minimum 11pt font. Longer answers will not be marked.]

e. (20 marks) Extend your bottom-up dynamic programming solution from part (c) to calculate an optimal schedule of activities (that produces the least loss) by implementing the public static method `optimalServicesDynamic` in the `Dynamic` class from the `assignment2` package.

   Like method `optimalLossDynamic`, your implementation of this method should run in polynomial time (in terms of k). It should be a bottom-up dynamic programming **(not memoised)** solution.

Practicalities

Do not change the class name of the `Recursive` or `Dynamic` classes or the package to which those files belong. You many not change the signatures of the methods that you have to implement in any way or alter their specifications. (That means that you cannot change the method name, parameter types, return types or exceptions thrown by the those methods.) Do not modify any of the other classes or interfaces or enumerated types defined in package `assignment2`.

You are encouraged to use Java 8 SE API classes, but no third party libraries should be used. (It is not necessary, and makes marking hard.) Don't write any code that is operating-system specific (e.g. by hard-coding in newline characters etc.), since we will batch test your code on a Unix machine. Your source file should be written using ASCII characters only.

You may not write and submit any additional classes. Your solution to Q1(a) should be self-contained in the `Recursive` class. Similarly your solution to parts Q1(c) and Q1(e) should be self-contained in the `Dynamic` class. Both of these classes will be tested in isolation and should not depend upon each other.

The zip file for the assignment also some junit4 test classes to help you get started with testing your code. The Junit4 test classes as provided in the package `assignment2.test` are not intended to be an exhaustive test for your code. Part of your task will be to expand on these tests to ensure that your code behaves as required.

Your programming implementations will be tested by executing our own set of junit test cases. Code that is submitted with compilation errors, or is not compatible with the supplied testing framework will receive 0 marks. A Java 8 compiler will be used to compile and test the code. The `Recursive` class will be tested in isolation from the `Dynamic` class.

Implementations that do not satisfy the assignment requirements will receive 0 marks even if they pass some of the test cases (e.g. if the solution given to Q1(c) is not a bottom-up dynamic programming solution, then it will receive 0 marks.)

You may lose marks for poorly structured, poorly documented or hard to comprehend code, or code that is not compatible with the assignment requirements. Line length should be less than or equal to 80 characters so that it can be printed – please use spaces to indent your code instead of tabs to ensure compatibility with different machines. Don't leave print statements in your submitted code.

Evaluation Criteria

Question 1

- **Question 1 (a) (20 marks)**

  Given that your implementation satisfies the requirements of the question, your implementation will be evaluated for correctness by executing our own set of junit test cases.

  20 : All of our tests pass

  16 : at least 80% of our tests pass

  12 : at least 60% of our tests pass

  8 : at least 40% of our tests pass

  4 : at least 20% of our tests pass

  0 : less than 20% of our test pass or work with little or no academic merit

  Note: Code that is submitted with compilation errors, or is not compatible with the supplied testing framework will receive 0 marks. A Java 8 compiler will be used to compile and test the code.

  Implementations that do not satisfy the assignment requirements will receive 0 marks even if they pass some of the test cases.

  The `Recursive` class will be tested in isolation from the `Dynamic` class.

- **Question 1 (b) (20 marks)**

  For this part of the question, the analysis should be no more than 1/2 of a page using minimum 11pt font. Longer solutions will receive 0 marks. Also, if a plausible, neat, legible and simple to understand solution to Q1(a) has not been given, this question will receive 0 marks. Otherwise the following marking criteria applies.

  20 : A correct asymptotic lower bound on the worst-case time complexity the recursive algorithm from Q1(a) is given in terms of parameter k. The lower bound, which should be exponential in k, should be as tight as reasonably possible for the algorithm at hand. The time-complexity given should be clearly justified by giving and solving a correct (lower bound) recurrence derived from your algorithm. Any assumptions made in the analysis are reasonable and clearly stated. Asymptotic notation should be used correctly and the asymptotic time complexity given has been simplified to remove lower order terms and unnecessary constant factors.

  15 : A correct asymptotic lower bound on the worst-case time complexity the recursive algorithm from Q1(a) is given in terms of parameter k. The lower bound should be exponential in k. The time-complexity given should be mostly clearly justified by giving and solving a correct (lower bound) recurrence derived from your algorithm. Any assumptions made in the analysis are mostly reasonable and clearly stated.

10 : A reasonable attempt has been made to give a tight asymptotic lower bound on the worst-case time complexity of the recursive algorithm from Q1(a) in terms of parameter k, and to justify it with respect to a recurrence derived from the algorithm, however the analysis or justification may contain minor mistakes or omissions or lack clarity.

5 : An attempt has been made to give an asymptotic lower bound on the worst-case time complexity of the recursive algorithm from Q1(a) in terms of parameter k, and justify it, however it contains either a major mistake or many mistakes, gives an unreasonably loose lower bound, or is not clearly justified by giving and solving a correct (lower bound) recurrence derived from your algorithm.

0 : Work with little or no academic merit.

- **Question 1 (c) (30 marks)**

  Given that your implementation satisfies the requirements of the question (i.e. it is a bottom-up dynamic programming (not memoised) solution that runs in polynomial time in terms of k), your implementation will be evaluated for correctness and efficiency by executing our own set of junit test cases.

  30 : All of our tests pass

  24 : at least 80% of our tests pass

  18 : at least 60% of our tests pass

  12 : at least 40% of our tests pass

  6 : at least 20% of our tests pass

  0 : less than 20% of our test pass or work with little or no academic merit

  Note: Code that is submitted with compilation errors, or is not compatible with the supplied testing framework will receive 0 marks. A Java 8 compiler will be used to compile and test the code.

  Implementations that do not satisfy the assignment requirements will receive 0 marks even if they pass some of the test cases.

  The `Dynamic` class will be tested in isolation from the `Recursive` class.

- **Question 1 (d) (10 marks)**

  For this part of the question, the analysis should be no more than 1/2 of a page using minimum 11pt font. Longer solutions will receive 0 marks. Also, if a plausible, neat, legible and simple to understand solution to Q1(c) has not been given, this question will receive 0 marks. Otherwise the following marking criteria applies.

  10 : A correct asymptotic upper bound on the worst-case time complexity of the algorithm from Q1(c) is given in terms of parameter k. The upper bound, which should be polynomial in k, should be as tight as reasonably possible for the algorithm at hand. The time-complexity given should be clearly justified with respect to the algorithm. Any assumptions made in the analysis are reasonable and clearly stated. Asymptotic notation should be used correctly and the asymptotic time complexity given has been simplified to remove lower order terms and unnecessary constant factors.

  7 : A correct asymptotic upper bound on the worst-case time complexity the algorithm from Q1(c) is given in terms of parameter k. The upper bound should be polynomial in k. The time-complexity given should be mostly clearly justified with respect to the algorithm. Any assumptions made in the analysis are mostly reasonable and clearly stated.

5 : A reasonable attempt has been made to give a tight asymptotic upper bound on the worst-case time complexity of the algorithm from Q1(c) in terms of parameter k, and to justify it, however the analysis or justification may contain minor mistakes or omissions or lack clarity.

3 : An attempt has been made to give an asymptotic upper bound on the worst-case time complexity of the algorithm from Q1(c) in terms of parameter k, and justify it, however it contains either a major mistake or many mistakes, gives an unreasonably loose lower bound, or is not clearly justified.

0 : Work with little or no academic merit.

- **Question 1 (e) (20 marks)**

  Given that your implementation satisfies the requirements of the question (i.e. it is a bottom-up dynamic programming (not memoised) solution that runs in polynomial time in terms of k), your implementation will be evaluated for correctness and efficiency by executing our own set of junit test cases.

  20 : All of our tests pass

  16 : at least 80% of our tests pass

  12 : at least 60% of our tests pass

  8 : at least 40% of our tests pass

  4 : at least 20% of our tests pass

  0 : less than 20% of our test pass or work with little or no academic merit

  Note: Code that is submitted with compilation errors, or is not compatible with the supplied testing framework will receive 0 marks. A Java 8 compiler will be used to compile and test the code.

  Implementations that do not satisfy the assignment requirements will receive 0 marks even if they pass some of the test cases.

  The `Dynamic` class will be tested in isolation from the `Recursive` class.

Change Log

- 7/10/2020 - Slight modification in the wording of the paragraph describing the `fullServiceCapacity` array to make it more clear how the i indices work. This change was made to make it more explicit that the indices of the array start from zero.