COMP4500/7500
Advanced Algorithms and Data Structures

School of Information Technology and Electrical Engineering
The University of Queensland, Semester 2, 2019

Assignment 2

Due at 4pm, Friday 18th of October 2019.
This assignment is worth 20% (COMP4500) or 15% (COMP7500) 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 written (not programming) questions (i.e. Q1(b), 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)

Suppose that you are running a small company for n consecutive (whole) days, day $0, 1, \ldots, n-1$.

During that time you have m job opportunities, where each *job* has a start day, an end day, and a payment (in whole dollars) that would be received for completing the job (no money is received for only partially completing a job). To complete a job, you must hire a worker to work on it full time for the start day of the job, the end day of the job, and all of the days in between the start and end day of the job.

For each of the job opportunities you are given, the start day is less than or equal to the end day, and the payment is greater than 0 dollars. Each of the job opportunities start no earlier than day 0 and end no later than the last day that you are running the company, day $n-1$.

Two job opportunities j_1 and j_2 are *compatible* if and only if they do not overlap (i.e. if either j_1 ends before j_2 starts, or j_2 ends before j_1 starts).

You only have one *worker* available to complete the job opportunities that you choose to accept. For $d \in \{0, 1, \ldots, n-1\}$, the cost of hiring the worker on day d is given by $cost[d]$, where *cost* is an array of n non-negative integers (indexed from 0).

During your time in charge of the company, you are able to hire the worker to work in shifts, where a *shift* has a start day and an end day (that is greater than or equal to the start day). For a given shift the worker must work full time for the start day of the shift, the end day of the shift, and all the days in between the start and end day of the shift.

Each of the shifts that you hire the worker for must start and end on days that you are in charge of the company, and must last for less than or equal to $maxShiftLength$ days. The shifts that you hire the worker on cannot overlap, and the worker must have at least $minShiftBreak$ free days between shifts (e.g. if a shift finishes on day d, then the earliest day that the next shift can start is day $d + minShiftBreak + 1$).

The *cost of hiring a worker for a shift* that starts on day s and ends on day f is $\sum_{d=s}^{f} cost[d]$, the sum of the costs of hiring the worker for all of the days that make up the shift.

A job *can be completed during a shift* if the start day of the job is greater than or equal to the start day of the shift, and the end day of the job is less than or equal to the end day of the shift.

A selected list of chosen shifts and a selected list of the chosen job opportunities, is said to be *valid* if and only if:

- Each chosen job is one of the job opportunities that you were given;
- The list of chosen jobs is ordered in ascending order of the end day of the jobs;
- All of the chosen jobs are compatible (i.e. they do not overlap);
- The list of chosen shifts is ordered in ascending order of the end day of the shifts;
- The chosen shifts cannot overlap, and the worker must have at least $minShiftBreak$ free days between shifts;
- Each of the chosen shifts must start and end on days that you are in charge of the company;
- Each chosen shift must last for less than or equal to $maxShiftLength$ days;
- Each of the chosen jobs must be able to be completed during one of the chosen shifts.

For a valid selection of shifts and jobs, the *profit earned by you for your company* is the sum of the payments you would receive for completing the chosen jobs, minus the cost of hiring the worker for all of the chosen shifts.

You are responsible for selecting a valid list of shifts and jobs that would result in the largest possible profit earned by you for your company.

The *maximum profit that can be earned by you for your company*, is defined to be the greatest profit earned by you for your company, given any valid selection of shifts and jobs.

The task: Given parameters,

- The array *cost* of worker's costs for each of the n days you are in charge of the company;
- The minimum number of days, $minBreakLength$, between worker's shifts;
- The maximum length, $maxShiftLength$, of the worker's shifts;
- The list of m jobs that are sorted in ascending order of their end day,

your task is to find the maximum profit that can be earned by you for your company.

Example As an example, consider the following scenario in which $n = 13$ and $m = 9$:

$$\begin{array}{lcl} costs & = & [1,2,2,2,3,0,5,1,3,2,2,3,1] \\ minShiftBreak & = & 2 \\ maxShiftLength & = & 6 \\ jobs & = & [(1,1,5),(0,2,15),(4,4,9),(2,5,17),(5,5,5),(5,6,12),(8,8,2),(11,12,5),(11,12,6)] \end{array}$$

where each job above is described first by its start day, then its end day, and then its payment.

There are many possible valid selections of jobs and shifts. For example,

$$\begin{array}{lcl} chosenShifts & = & [(0,5),(11,12)] \\ chosenJobs & = & [(0,2,15),(4,4,9),(5,5,5),(11,12,6)] \end{array}$$

is a valid selection with profit:

$$profit(chosenShifts, chosenJobs) \;=\; (15+9+5+6) - ((1+2+2+2+3+0)+(3+1)) \;=\; 21$$

Another example of a valid selection is:

$$\begin{array}{lcl} chosenShifts & = & [(1,5)] \\ chosenJobs & = & [[(1,1,5),(4,4,9)] \end{array}$$

which has profit:

$$profit(chosenShifts, chosenJobs) \;=\; (5+9) - (2+2+2+3+0) \;=\; 14-9=5$$

In this example, the maximum profit that can be earned by you for your company is 21.

a. (20 marks) Implement the public static method maximumProfitRecursive 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 maximum profit that can be earned by you for your company. To implement that method, you will need to provide an implementation for the private static method maximumProfitRecursive from the same class.

   The recursive solution does NOT need to find a valid selection of shifts and jobs that would result in the largest possible profit earned by you for your company – it just needs to determine the maximum profit. Efficiency is not at all a concern for this part, so focus on an elegant solution. (You must not provide a dynamic programming solution to this question.)

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 job opportunities is m, and the number of days you are running the company is n, give an asymptotic lower bound on the worst-case time complexity of your recursive algorithm in terms of parameters m and n. Make your bound as tight as possible.

   As part of your answer you need to provide a lower-bound recurrence (in terms of parameters m and n) that describes the worst-case time complexity of your recursive algorithm; you need to justify your recurrence, explaining why it appropriately describes a lower bound on the worst-case time complexity of your algorithm; and you need to solve your recurrence (showing your working) to derive your answer to this question.

   [Make your answer as concise as possible – it should be no more than 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 maximumProfitDynamic in the Dynamic class from the assignment2 package that accompanies this handout.

   Your dynamic programming solution should run in polynomial time (in terms of m and n).

   The dynamic solution does NOT need to find a valid selection of shifts and jobs that would result in the largest possible profit earned by you for your company – it just needs to determine the maximum profit.

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 parameters m (the number of job opportunities) and n (the number of days you are running the company). Make your bounds as tight as possible and justify your solution.

   [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 a valid selection of shifts and job opportunities that results in the largest possible profit to your company, by implementing the public static method optimalSolutionDynamic in the Dynamic class from the assignment2 package.

   Like method maximumProfitDynamic, your implementation of this method should run in polynomial time (in terms of m and n). It must 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 compatability 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 one 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 parameters m and n. The lower bound, which should be exponential in m, should be as tight as reasonably possible for the algorithm at hand. The time-complexity given should be clearly justified by giving, justifying 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 parameters m and n. The lower bound should be exponential in m. The time-complexity given should be mostly clearly justified by giving, justifying 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 parameters m and n, 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 both give an asymptotic lower bound on the worst-case time complexity of the recursive algorithm from Q1(a) in terms of parameters m and n, and to justify it in terms of a recurrence derived from your algorithm, however it contains either a major mistake or many mistakes, gives an unreasonably loose lower bound, or is not clearly justified by giving, justifying 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 m and n), 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 parameters m and n. The upper bound, which should be polynomial in m and n, 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 parameters m and n. The upper bound should be polynomial in m and n. 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 parameters m and n, 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 parameters m and n, and justify it, however it contains either a major mistake or many mistakes, gives an unreasonably loose upper 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 m and n), 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.