Semester 2 2017
COMP3702/7702 ARTIFICIAL INTELLIGENCE
ASSIGNMENT 3: MDP-based Investment Strategy

Note:

- This assignment consists of two parts: Programming and report.
- You can do this assignment in a group of at most 3 students. This means you can also do the assignment individually.
- For those who choose to work in a group:
  - All students in the group must be enrolled in the same course code, i.e., all COMP3702 students or all COMP7702 students. Please register your group name in http://goo.gl/mzX66C before **11.59pm on Monday, 27 October 2017**. If you have not registered your group by the said time, you will need to work on the assignment individually.
  - All group members are expected to work in both programming and report. In the report, you are required to write the role of each team member.
- Submission Instruction:
  - You must write your program in either Java or C/C++ or Python. Supporting code will be given only for Java.
  - Your program should compile from command prompt (e.g., using ant or make/cmake). It should generate an executable named a3-[courseCode]-[ID] that can be run from command prompt as:

    > [OptionalAdditionalCmd] a3-[courseCode]-[ID] inputFileName outputFileName

    [courseCode] must be replaced with 3702 or 7702, depending on which class you are enrolled in. If you work individually, please replace [ID] with your student ID. Otherwise, please replace [ID] with your group name. [OptionalAdditionalCmd] must be replaced with python or java –jar or ignored.The report should be in .pdf format and named a3-[courseCode]-[ID].pdf.
  - The report and all the source codes necessary to compile your program should be placed inside a folder named a3-[courseCode]-[ID]. Please submit only source codes (i.e., remove all object files).
  - The folder should be zipped under the same name, i.e., a3-[courseCode]-[ID].zip, and the zip file should be submitted via turnitin before **11.59pm on Friday, 17 November 2017.** Of course, you are welcome to submit early.

With the start-up boom, it is no longer that rare for an individual to own multiple businesses. To help such individuals managed their businesses, UQ-Invest is developing a new product to help individuals decide how to divide his/her funding to multiple ventures, on a fortnightly basis. The goal is of course, to make the most profit in the long run. You have been hired to develop a simplified prototype for such a software system.

Your task is to create a system that uses historical profit data from the ventures, to decide how funding should be divided among an individual's multiple ventures, so that he/she makes the most profit in the long run. In this assignment, we assume that the software focuses on funding to manufacture the product of the business venture. The rest of this

document describes the detail of the problem.

Each user of the software system can be classified into either bronze, silver, gold, or platinum user, based on the number of bussiness ventures he/she owns and the total amount of funding that he/she can provide every fortnight. The details are:

- Bronze user. The user owns 2 business ventures. The total reserved funding for manufacturing products is $30k. At the beginning of each fortnight, the user can provide at most a total of additional $30k funding.
- Silver user. The user owns 2 business ventures. The total reserved funding for manufacturing products is $50k. At the beginning of each fortnight, the user can provide at most a total of additional $40k funding.
- Gold user. The user owns 3 business ventures. The total reserved funding for manufacturing products is $60k. At the beginning of each fortnight, the user can provide at most a total of additional $40k funding.
- Platinum user. The user owns 3 business ventures. The total reserved funding for manufacturing products is $80k. At the beginning of each fortnight, the user can provide at most a total of additional $50k funding.

To simplify the problem, the funding is **always given and used as a multiple of $10,000**. For instance, for a bronze user, he/she can only provide total additional funding of $0, $10,000, $20,000, and $30,000 per fortnight. For compactness, we will remove the 4 zeros and just write the multiplier. For instance, in the above example, the total additional funding is written as 0, 1, 2, and 3 to represent $0, $10,000, $20,000, and $30,000.

Each business venture will use the funding to manufacture its product whenever a customer has agreed to purchase its product. We assume that a unit of the product will cost $10,000 to manufacture and can be sold for a multiple of $10,000 (which varies from venture to venture – see the input file format). For each sale of a product, the venture keeps 40% of the sale payment, used to cover its operational costs (this amount cannot be put back into the cash reserved for manufacturing products) and passes the remaining 60% to the owner of the venture. Whenever the venture fails to provide the product (due to lack of manufacturing fund), the owner has to pay a penalty of 25% of the selling price of the product.

To simplify the problem further, UQ-Invest assumes:

- There will be no error in the manufacturing process.
- Each business venture manufactures only one type of product.
- Since the financial health depends on its manufacturing fund and are made public, the number of orders received by one venture depends only on the amount of manufacturing fund the venture has, right after the most recent addition. It is independent of the manufacturing fund other ventures have.
- If the venture ran out of manufacturing funds, the purchase will fail and therefore the venture and its owner will not receive any payment.
- The performance of the store is measured fortnightly.
- The system should assume that the venture will remain open forever.

Although for simplicity in computing the actual profit gained, the system will be tested on a finite number of fortnights.

Given the type of user, the stochastic model of the number of orders to each venture, and information about the available manufacturing fund, you need to develop a system that identifies how much funding should be given to which business ventures, so that the total expected (return minus penalty) is maximized.

What you need to do

Your tasks in this assignment can be classified into 4 parts:

1. Design the solution. This task contains two main components, i.e., framing the problem as an MDP problem and deciding how to solve the problem. Note that you should design the MDP components, i.e., what are the states, actions, etc., manually. However, the exact parameters of your MDP problem will depend on the given input file.
2. Implement your design. Your program is allowed a maximum of 3 minutes computation prior to each simulation run. If you use an online method for solving MDP, at each step during run-time, your program is allowed a maximum of 30 seconds computation time. Your program will be run on a PC with the same specification as those in the tutorial rooms. The time requirement is for a program that runs as a single-threaded process. If you use multi-threading, then we will divide the aforementioned time limit with the number of threads you use. Note: ***You are not allowed to use any library for linear algebra, optimization, and MDP solver***.
3. Implement the basic synchronous value iteration algorithm in order to compare your design with this baseline (at least for the bronze and silver users).
4. Write a report. Similar to the past two assignments, you only need to answer the questions we post for the Report (see the last part of this document). However, to answer these questions well, you do need to do #3.

Input and Output format

Input format. The input is a single .txt file, containing information about the ventures the user owns: the selling price of each venture's products, the manufacturing fund available to each venture (as part of the total manufacturing fund that the user has), and the behaviour of the venture's customers.
The format of the input file is as follows.

1. The first line is the user's type, i.e., bronze, silver, gold, or platinum.
2. The second line is the discount factor.
3. The third line is the number of fortnight that the system will be tested.
4. We assume that the ventures a users owns are indexed from 1 to V, where V is the number of ventures. The fourth line then consists of V positive integers, separated by a white space. The first integer represents the selling price of the product manufactured by venture-1, the second represents the selling price of the product manufactured by venture-2, etc.
5. The fifth line represents the manufacturing fund that each venture has at the beginning of the the first fortnight. This line consists of V

positive integers, where the first integer represents the manufacturing fund (in multiples of $10,000) of venture-1, the second represents the manufacturing fund of venture -2, etc.

6. The subsequent lines represent the estimated number of orders, represented as V probability matrices. Each matrix represents the order estimates for a particular business venture. Suppose P_i is the probability matrix that corresponds to venture-i ($i \in [1, V]$). Then, the element e_{jk} at row-j ($j \in [0, M]$) and column-k ($k \in [0, M]$) of P_i represents the conditional probability that in a fortnight, the total number of orders to venture-i is k, given there is j*$10,000 of manufacturing funding available for venture-i at the beginning of the fortnight. The maximum value of the manufacturing fund is M*$10,000. Note that the indexing for the matrices starts from 0 (i.e., to represent zero order and fund), which means each matrix is of size $(M+1)^2$.
   The list of probability matrices are written in the input file, starting at line-6:
   Line-6 to line-(6+M): represent the rows of P_1. The numbers in each line (i.e., the columns) are separated by a white space. Each probability value has at most 3 decimal digits.
   :
   :
   Line-(6+(M+1).(V-1)) to line-(5+(M+1).V): represent the rows of P_V in a sequential order.

Our supporting code will provide a simulator for order requests, collated into a fortnightly order.

Output format. Your program should output the state, the number of orders, and the additional manufacturing funding given to the ventures every fortnight. The format is as follows:

1. The first line is the number of fortnight the system is being tested. Let's denote this as N. The output file consists of $2N$+2 lines.
2. Line-2 represents the manufacturing fund at the beginning of the first fortnight. It consists of V positive integers, where each integer is separated by a white space. The ith integer represents the fund for venture-i (in multiples of $10,000).
3. Line-3 represents the number of order for fortnight-1. It consists of V positive integers, where each integer is separated by a white space. The ith integer represents the number of orders to venture-i.
4. Line-4 represents additional manufacturing fund provided at the end of fortnight-1. It consists of V positive integers, where each integer is separated by a white space. The ith integer represents the additional fund provided to venture-i.
5. Line-5 represents the number or orders for fortnight-2. It consists of V positive integers, where each integer is separated by a white space. The ith integer represents the number of orders to venture-i.
6.
   :
   :
7. Line-2N represents additional manufacturing fund provided at the beginning of fortnight-(N-1). It consists of V positive integers, where each integer is separated by a white space. The ith integer represents the additional fund for venture-i.

8. Line-2*N*+1 represents the number of orders for fortnight-*N*. It consists of V positive integers, where each integer is separated by a white space. The i^{th} integer represents the number of orders to venture-i.
9. Line-2*N+2* represents additional manufacturing fund provided at the beginning of fortnight-(*N)*. It consists of V positive integers, where each integer is separated by a white space. The i^{th} integer represents the additional fund for venture-i.

Examples of the input and output files are in the accompanying supporting software.

Grading for the Programming Part (total points: 60/100)

Your program is deemed to solve the problem successfully when the system uses less than or equal time than the maximum allowable time, and the customer earns a total of more than 0.5**M***N*, where *M* is the maximum manufacturing fund that the user can provide per fortnight and *N* is the number of fortnight that the inventory system will be tested. If you use sampling-based method, we will run your program 10X for each problem. Your program is deemed to solve the problem if it solves at least 5 out of the 10 runs.

The details of the grading scheme is as follows:

COMP3702:

- >= 1 & < 10: The program does not compile nor run.
- >= 10 & < 30: The program runs but fails to solve any problem.
- >= 30 & < 40: The program solves at least one of the problems for the bronze user. There will be two bronze users, and each problem is worth 5 points.
- >= 40 & < 50: The program solves at least one of the problems for the silver user. There will be two silver users, and each problem is worth 5 points.
- >= 50 & <= 60: The program solves at least one of the problems for the gold user. There will be three gold users, and each problem is worth 5 points.

COMP7702:

- >= 1 & < 10: The program does not compile nor run.
- >= 10 & < 20: The program runs but fails to solve any problem.
- >= 20 & < 30: The program solves at least one of the problems for the bronze user. There will be two bronze users, and each problem is worth 5 points.
- >= 30 & < 40: The program solves at least one of the problems for the silver user. There will be two silver users, and each problem is worth 5 points.
- >= 40 & < 50: The program solves at least one of the problems for the gold user. There will be two gold users, and each problem is worth 5 points.

- >= 50 & <= 60: The program solves at least one of the problems for the platinum user. There will be three platinum users, and each problem is worth 5 points.

If your situation satisfies more than one marking band, we will use the higher band.

Report (total points: 40/100)

Your report must contain answers to the following questions:

1. [10 points] Please define your MDP problem.
2. [10 points] Please explain the method you use to solve the problem.
3. [20 points] Please analyse your algorithm's accuracy and speed, via comparison study. For this purpose, at the very least, you need to compare your method against the base value iteration with synchronous update (as described in class) on a bronze and silver user problems. If the algorithm you implement is the aforementioned basic value iteration, please explain how you would modify it to solve the harder problems.

Please note that in each of the above question, when explanation is requested, the explanation part is 80% of the total points. For the explanation part, you will get higher mark for stronger argument. A strong argument should at least include a logical explanation of why and include experimental results to back your explanation. Please also note that good explanation is NOT equal to long explanation!!!

oOo That's All, Folks oOo