Xi'an Jiaotong-Liverpool University
西交利物浦大学

XJTLU Entrepreneur College (Taicang) Cover Sheet

Module code and Title	**DTS203TC Design and Analysis of Algorithms**
School Title	**School of AI and Advanced Computing**
Assignment Title	**Coursework**
Submission Deadline	**Sunday, March 24th 23:59 (UTC+8 Beijing), 2024**
Final Word Count	
If you agree to let the university use your work anonymously for teaching and learning purposes, please type **"yes"** here.	

I certify that I have read and understood the University's Policy for dealing with Plagiarism, Collusion and the Fabrication of Data (available on Learning Mall Online). With reference to this policy I certify that:

- My work does not contain any instances of plagiarism and/or collusion.
  My work does not contain any fabricated data.

By uploading my assignment onto Learning Mall Online, I formally declare that all of the above information is true to the best of my knowledge and belief.

Scoring – For Tutor Use	
Student ID	

Stage of Marking	**Marker Code**	**Learning Outcomes Achieved (F/P/M/D) (please modify as appropriate)**			**Final Score**
		A	**B**	**C**	
1st Marker – red pen					
Moderation – green pen	**IM Initials**	The original mark has been accepted by the moderator (please circle as appropriate):			Y / N
		Data entry and score calculation have been checked by another tutor (please circle):			Y
2nd Marker if needed – green pen					

For Academic Office Use			**Possible Academic Infringement (please tick as appropriate)**	
Date Received	**Days late**	**Late Penalty**	☐ **Category A**	Total Academic Infringement Penalty (A,B, C, D, E, Please modify where necessary) _____
			☐ **Category B**	
			☐ **Category C**	
			☐ **Category D**	
			☐ **Category E**	

Xi'an Jiaotong-Liverpool University
西交利物浦大学

DTS203TC Design and Analysis of Algorithms

Coursework

Deadline: Sunday, March 24th 23:59 (UTC+8 Beijing), 2024

Percentage in final mark: 40%

Learning outcomes assessed:

A. Describe the different classes of algorithms and design principles associated with them; Illustrate these classes by examples from classical algorithmic areas, current research and applications.
B. Identify the design principles used in a given algorithm, and apply design principles to produce efficient algorithmic solutions to a given problem.
C. Have fluency in using basic data structures in conjunction with classical algorithmic problems.

Late policy: 5% of the total marks available for the assessment shall be deducted from the assessment mark for each working day after the submission date, up to a maximum of five working days

Risks:

- Please read the coursework instructions and requirements carefully. Not following these instructions and requirements may result in loss of marks.
- The assignment must be submitted via Learning Mall to the correct drop box. Only electronic submission is accepted and no hard copy submission.
- All students must download their file and check that it is viewable after submission. Documents may become corrupted during the uploading process (e.g. due to slow internet connections). However, students themselves are responsible for submitting a functional and correct file for assessments.
- Academic Integrity Policy is strictly followed.
- The use of Generative AI for content generation is not permitted on this assessed coursework. Submissions will be checked through Turnitin.

Overview

In this coursework, you are expected to design and implement algorithms to produce solutions to four given problems (Tasks 1-4) in Python. For Tasks 1-4, you should have function(s) to receive task input as parameters, implement your algorithm design and return results. You also need to write a short report answering a list of questions in Task 5 that are related to the given four problems.

Xi'an Jiaotong-Liverpool University
西交利物浦大学

Task 1 (15 marks)

You have *n* coins and a balance. Among the coins, there is one fake coin which has different weight than the other coins. All the coins, except the fake coin, have exactly the same weight. Utilizing the balance, you can compare the weight of any number of coins on each pan. The balance will indicate whether the set of coins on one pan weighs the same as the set on the other pan. Assume the number of coins is a power of 3 ($n = 3^k$). Implement an efficient algorithm to find the fake coin.

Input: coins = [10, 10, 10, 9, 10, 10, 10, 10, 10]

Output: 4

Explanation: In the given example, there are 9 coins, with the genuine coins weighing 10 units each and the fake coin weighing 9 units. The fourth (output) coin is identified as the fake coin.

You should have a function named findFakeCoin to receive a list of integers and return the index of fake coin (int). Please consider the **time complexity** when you design your algorithm. A naïve approach will result in loss of marks.

Task 2 (15 marks)

Consider a *d*-ary heap as a generalization of the binary heap, where each node has *d* children instead of 2. Implement a *d*-ary max heap and performs heap sort on a given array.

Example:

Input: nums = [7, 6, 5, 9, 8], d = 3

Output: [5, 6, 7, 8, 9]

You should create a function named dHeapSort that takes a list of integers to be sorted and an integer *d* indicates *d*-ary as parameters. The function should return a *list* containing the sorted integers based on the *d*-ary heap sort algorithm.

Task 3 (15 marks)

You're driving an electric car from Shanghai to Beijing, where there are charging stations along the way at distances $x_1, x_2, \ldots, x_n$ from Shanghai. Due to varying wait times *c* and charge speeds *g*, charging *k* kilometers worth of electric at station x_i takes $c_i + kg_i$ minutes. Your car has a capacity sufficient to travel 400 kilometers on a single charge. Assume car battery begin with 0 at the first station x_1 in Shanghai and x_n is your destination in Beijing. Design an efficient algorithm that finds where you should stop to spend the minimum amount of time at charging stations during your trip.

Example:

Input: x = [0, 100, 300, 600, 800, 1000], c = [0, 10, 0, 20, 10, 0], g = [0.05, 0.2, 0.1, 0.2, 0.1, 0]

Output: [20,0,30,40,30,0]

Explanation: In the provided example, there are a total of 6 stations. The objective is to drive from the first station to the final one, optimizing the time spent at each station. The output details the time spent at each station, aiming to minimize the overall time during the trip. For instance, the time spent at the first station is calculated as 0 + 400 * 0.05 = 20 minutes.

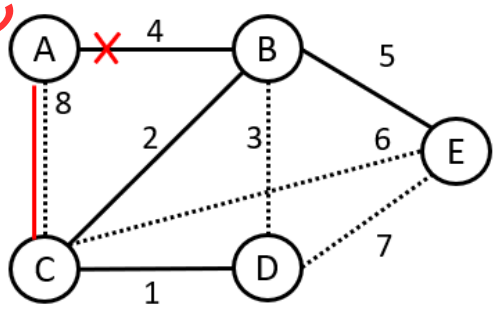
You should have a function named timeSpent to receive the information of distance *x* (List[int]), wait time *c* (List[int]), charge speed *g* (List[int]) and return the time spend at each station *t* (List[int]). Please consider the **time complexity** when you design your algorithm. A naïve approach will result in loss of marks.

Task 4 (15 marks)

Consider an undirected graph G = (V, E) with distinct weights assigned to each edge. Your task is to find the minimum spanning tree (MST), denoted as T, on G. Anticipating the potential removal of one edge in the future, implement an efficient algorithm to compute the MST T along with a backup edge, denoted as r(e), for each edge e in T. This ensures that if any edge e is removed, adding its corresponding backup edge r(e) to the tree will quickly create a new minimum spanning tree in the modified graph.

Example:

Input: graph = {'A': {'B': 4, 'C': 8},
'B': {'A': 4, 'C': 2, 'D': 3, 'E': 5},
'C': {'A': 8, 'B': 2, 'D': 1, 'E': 6},
'D': {'B': 3, 'C': 1, 'E': 7},
'E': {'B': 5, 'C': 6, 'D': 7}}



Output: {('A', 'B'): ('A', 'C'), ('B', 'C'): ('B', 'D'), ('C', 'D'): ('B', 'D'), ('B', 'E'): ('C', 'E')}

Explanation: The input graph (as shown in figure) can be represented as a dictionary where the keys are vertices, and the values are dictionaries representing the edges going out of that vertex and the weights of those edges. The output can also be represented as a dictionary where the keys are the edges of the minimum spanning tree T, and the values are the corresponding backup edge of each edge in T. For example, if edge ('A', 'B') is removed from the graph G, adding backup edge ('A', 'C') will form a new minimum spanning tree.

You should have a function named modifiedMST to receive the graph represented as a dictionary and return the minimum spanning tree T with backup edges represented as a dictionary. Please

consider the **time complexity** when you design your algorithm. A naïve approach will result in loss of marks.

Task 5 (40 marks)

Answer the following questions in your report. Maximum 800 words for Task 5. (Clarity and brevity are valued over length).

T5-1: For Task 1, give a recurrence of the running time of your algorithm and solve the recurrence with two different methods.

T5-2: For Task 2, what is the running time of d-ary heap's insert and heapify operations? Will d-ary heap sort faster than binary heap sort? Justify your answer.

T5-3: For Task 3, explain the design, show the correctness, and analyse the time and space complexity of your algorithm.

T5-4: Given the same graph G from Task 4, is it possible to have more than one minimum spanning tree T? Justify your answer.

Submission

Electronic submission on Learning Mall is mandatory. You need to submit a zip file (named DTS203TC-CW-YOUR_NAME.zip) containing the following documents.

1. Cover letter with your student ID.
2. Your source code for Tasks 1-4: *Solutions.ipynb*
3. A pdf file contains all the source code (should be the same as the submitted ipynb file) and your report (task 5). You can also write the report in jupyter notebook and export as a pdf file.

Generic Marking Criteria

Grade	Point Scale		Criteria to be satisfied
A	81+	First	➢ Outstanding work that is at the upper limit of performance. ➢ Work would be worthy of dissemination under appropriate conditions.

			➢ Mastery of advanced methods and techniques at a level beyond that explicitly taught. ➢ Ability to synthesise and employ in an original way ideas from across the subject. ➢ In group work, there is evidence of an outstanding individual contribution. ➢ Excellent presentation. ➢ Outstanding command of critical analysis and judgment.
B	70 - 80	First	➢ Excellent range and depth of attainment of intended learning outcomes. ➢ Mastery of a wide range of methods and techniques. ➢ Evidence of study and originality clearly beyond the bounds of what has been taught. ➢ In group work, there is evidence of an excellent individual contribution. ➢ Excellent presentation. ➢ Able to display a command of critical thinking, analysis and judgment.
C	60 - 69	Upper Second	➢ Attained all the intended learning outcomes for a module or assessment. ➢ Able to use well a range of methods and techniques to come to conclusions. ➢ Evidence of study, comprehension, and synthesis beyond the bounds of what has been explicitly taught. ➢ Very good presentation of material. ➢ Able to employ critical analysis and judgement. ➢ Where group work is involved there is evidence of a productive individual contribution

D	50- 59	Lower Second	➢ Some limitations in attainment of learning objectives but has managed to grasp most of them. ➢ Able to use most of the methods and techniques taught. ➢ Evidence of study and comprehension of what has been taught ➢ Adequate presentation of material ➢ Some grasp of issues and concepts underlying the techniques and material taught. ➢ Where group work is involved there is evidence of a positive individual contribution.
E	40 - 49	Third	➢ Limited attainment of intended learning outcomes. ➢ Able to use a proportion of the basic methods and techniques taught. ➢ Evidence of study and comprehension of what has been taught, but grasp insecure. ➢ Poorly presented. ➢ Some grasp of the issues and concepts underlying the techniques and material taught, but weak and incomplete.
F	0 - 39	Fail	➢ Attainment of only a minority of the learning outcomes. ➢ Able to demonstrate a clear but limited use of some of the basic methods and techniques taught. ➢ Weak and incomplete grasp of what has been taught. ➢ Deficient understanding of the issues and concepts underlying the techniques and material taught. ➢ Attainment of nearly all the intended learning outcomes deficient.

			➢ Lack of ability to use at all or the right methods and techniques taught. ➢ Inadequately and incoherently presented. ➢ Wholly deficient grasp of what has been taught. ➢ Lack of understanding of the issues and concepts underlying the techniques and material taught. ➢ Incoherence in presentation of information that hinders understanding.
G	0	Fail	➢ No significant assessable material, absent, or assessment missing a "must pass" component.

Marking Criteria

Tasks	100	Components	Description	Maximum Credit	Mark
Task 1	15	Implementation 6 marks	Correct function definition [0/1 mark]	1	
			Correct algorithm design [0/2 marks]	2	
			Algorithm implementation [0-3 marks]	3	
		Evaluation 8 marks	Time complexity [0/3 marks]	3	
			5 test cases will be used to evaluate the correctness of the function. 1 mark for each test case.	5	
		Code quality 1 mark	Readability, Formatting, Comments	1	
Task 2	15	Implementation 9 marks	Correct function definition [0/1 mark]	1	
			Data structure implementation [0/2/4 marks]	4	
			Algorithm implementation [0/2/4 marks]	4	
		Evaluation 5 marks	5 test cases will be used to evaluate the correctness of the function. 1 mark for each test case.	5	
		Code quality 1 mark	Readability, Formatting, Comments	1	
Task 3	15	Implementation	Correct function definition [0/1 mark]	1	

		6 marks	Correct algorithm design [0/2 marks]	2	
			Algorithm implementation [0-3 marks]	3	
		Evaluation 8 marks	Time complexity [0/3 marks]	3	
			5 test cases will be used to evaluate the correctness of the function. 1 mark for each test case.	5	
		Code quality 1 mark	Readability, Formatting, Comments	1	
Task 4	15	Implementation 6 marks	Correct function definition [0/1 mark]	1	
			Correct algorithm design [0/2 marks]	2	
			Algorithm implementation [0-3 marks]	3	
		Evaluation 8 marks	Time complexity [0/3 marks]	3	
			5 test cases will be used to evaluate the correctness of the function. 1 mark for each test case.	5	
		Code quality 1 mark	Readability, Formatting, Comments	1	
Task 5	40	Task 5-1 9 marks	Recurrence of the algorithm is correct [0/3 marks]	3	
			Solve recurrence with method 1 [0/3 marks]	3	
			Solve recurrence with method 2 [0/3 marks]	3	
		Task 5-2 9 marks	Running time of insert operation [0-3 marks]	3	
			Running time of heapify operation [0-3 marks]	3	
			Analysis of d-ary heap sort [0-3 marks]	3	
		Task 5-3 9 marks	Algorithm design [0/2 marks]	2	
			Correctness [0/3 marks]	3	
			Time and space complexity [0/2/4 marks]	4	
		Task 5-4 9 marks	'Yes/No' answer correct	3	
			Number of minimum spanning tree on G is well justified [0-6 marks]	6	
		Report quality 4 marks	Fluency and readability [0/2 mark] Formatting and conciseness [0/2 mark]	4	
			Late Submission?	☐ Yes ☐ No	Days late
			Final Marks		