Data Structure and Algorithms Universidad Carlos III de Madrid Case Study: COURSE 2019-2020	uc3m | Universidad Carlos III de Madrid

Application to manage product deliveries at Amazon.

Phase 2 - Trees

During phase 1 we built a set of data structures to manage package deliveries at Amazon.

As you already know, in order to assign a package to an active delivery staff member it was necessary to go through the list of people to find an active deliveryman within the proper zone. This operation has a complexity O(n), where n is the number of delivery people. Thus, in the worst case it will be necessary to traverse the whole list to either find a proper deliveryman or realize that there's no one to deliver

The aim of this phase is to develop a data structure which allows this kind of zone-search with logarithmic complexity. This structure must allow adding or removing zones aswell, so we can modify the planification as needed. Note that each zone can be assigned to one or more delivery people.

During this phase we're only focusing on modelling zones and their assigned people. We're NOT implementing any kind of functionality related to package delivery, since it's supposed to be done within phase 1.

Specifically, you need to develop a data structure containing the following methods:

- Implement a function, **createZone()**, which allows adding a new zone without people assigned to the data structure. Note that complexity must be logarithmic.
- Implement a function, **showZones()**, which shows all the zones ordered with ascending order. For each zone you will need to show the list of assigned people. Note that complexity must be logarithmic.
- Implement a method, **deleteZone()**. Given a zone, this method must remove it from the data structure and reassign its delivery people to remaining ones **in a balanced way**.
    - If a zone contains k delivery people, k/2 must be moved to previous zones and k/2 to the following ones.
    - You must distribute each k/2 people as balanced as possible among the zones.
- Implement a function, **showDistributors()**, that receives a zone and displays its distributors in alphabetical order.
- Implement a function, **assignsDistributor()**, that allows a distributor to be assigned to a given zone.
- Implement a function, **deleteDistributor()**, that deletes a distributor from a certain zone.
- Implement a function, **isBalanced()**, that checks if zones are stored in a balanced way in the data structure. You must base on the perfect balance, where for a given zone, the difference between the number of previous zones and next zones, must be 1 as maximum. If the structure is not balanced, then you must balance it following the perfect balancing strategy (or in size) explained in class. What advantages does the structure balancing offer?
- Create a **test** function that allows you testing each of the functions described above.

Complexity Analysis: At the beginning of each function you must include a comment indicating its complexity, the worst and the best cases.

Rules:

1. Using Python structures such as (Lists, Queues or Dictionaries, etc.) is not allowed.
2. However, you can use the implementation (linked lists and trees) that we have studied in class.
3. The case study must be carried out by a group of two members (both must belong to the same teaching group). Groups of more than two members are no allowed. In addition, Individual groups are not allowed since one of the competences to be evaluated will be teamwork. If you don't have a partner, please send an email to your practice teacher.
4. Delivery method: a task entitled “Amazon 2020 Case Study” will be posted on the master group.
5. Deadline: May 4, at 9.00 a.m.
6. Delivery format: a zip file with the python (.py) scripts of the three phases. The name of the zip file must contain the two NIAs of the students, separated by a hyphen. For example, 10001234_0005678.zip. Only one of the two members will be responsible for uploading the group solution.
7. Defence: during the class of Monday, May 4. The defence of the case study is an oral exam. Attendance is mandatory. If a student does not attend, his or her grade in the practical case will be 0. During this defence, the teacher will ask each team member a series of questions to be answered individually. Each team member should be able to discuss any of the decisions made in the design and implementation of any of the functionalities described in the case study. The final grade of the case study will be conditioned by the grade obtained in the defence. If a student is not able to discuss and defend certain decisions, they will not be qualified, even if they have been correctly implemented.
8. it is recommended to follow the recommendations described in Zen of Python (https://www.python.org/dev/peps/pep-0020/) and the style guide (https://www.python.org/dev/peps/pep-0008/) published on the official Python website.
9. Your solution must be divided into subprograms (modules) in order to make it more readable, easier to debug and maintain. Each module or subprogram must fulfil a well-defined task, and some modules need others to operate.