**(S2-20\_DSECLZG519) (Data structures and Algorithms Design)**

**Academic Year 2020-2021**

Software Design Documentation

**Assignment 2 PS5 – Assignment - Group 21**

**Overview:**

## This problem is about a maximizing the profit by correctly scheduling the problem using greedy method

**Given Problem Statement:**

A teacher has given a set of questions to the class. Each question takes a day to finish the task. Each problem has a deadline, if finished before the deadline the class gets extra bonus marks. No Bonus,If the problem is not finished before the deadline

For example, the class has been given 3 problems: [A,B,C]. The corresponding deadlines are [1,2,1] and the bonus for finishing the problem before time is [10,20,30]. For maximizing the bonus marks you can get problem C and B. While missing the deadline of problem A, which gives you a maximum 50 bonus marks earned. The order of the tasksis C-B-A

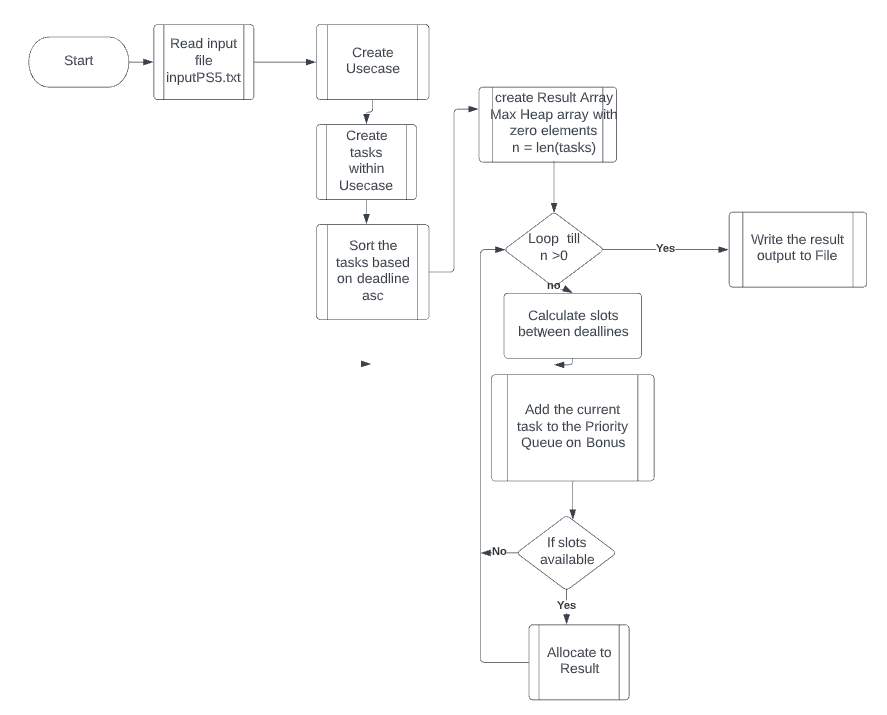


**Goal:**

The goal is to create a python program to read the list of use cases (problems) with various tasks and each task consist deadline and bonus, apply the greedy algorithm to get the maximum profit and then write the profit earned and order in which the tasks are taken to an output file

**Flow Diagram:**

The following diagram shows the overall flow



**Design + Data Structures:**

This program can be solved various ways. Our primary focus was to reduce the complexity of the program. In order to apply the sort on only bonus and try to allocate the tasks based on deadline then it ends up n^2 complexity. i.e, because sorting takes n\*log(n) and then looping through the tasks and assigning the tasks (can be n) based on the deadline availability (deadline can be n) makes it n\*n = n^2. Over all it is nlogn + n^2 ~= n^2.

We wanted to reduce this to n\*log(n) and hence we tried an approach where the first sorting based on deadline ascending and second sorting is based on bonus descending and then calculate the slots between the deadlines and allocate the task.

This algorithm uses list and three custom class one to hold attributes of task and one to hold usecases and encapsulate its functions

* TaskHeapQueue (Priority queue): This holds all methods to apply heapify (shift up /shift down) algorithms on the input list of tasks which eventually helps in order the data based on the bonus. It consist of four methods
  + Insert
  + Remove
  + Shiftup
  + Shiftdown
* Task: The custom class consists of three attributes
  + Name of the task, generated at runtime and value can be task:1, task:2, task:3 etc
  + Task deadline: positive integer
  + Task bonus : positive integer
* Usecase: Custom class consists of the following attributes and members
  + Attribute: Use case name : Generated at runtime based on the number of use case passed in the file
  + Attribute:
    - Tasks : List of tasks under a usecase
    - Assigned Tasks: Final results after applying the greedy algorithm
  + Method:
    - maximizeBonus: greedy algorithm implemented as part of this method to maximize the bonus
* List: This list holds list of use cases
* There are static method to implement the merge sort

This is the simplest data structure meeting all our requirements.

Note: The program didn’t use any python librabries other than lists and custom classes

**Run Time Analysis:**

The max complexity to solve one usecase is n\*log(n)

* Step-1 is sort the tasks in ascending order w.r.t deadline. This is implemented using merge sort and it takes O(n\*(log n)) complexity
* Step-2: It then uses priority queue (simple implementation using TaskHeapQueue) to push all elements based on the priority and in this case priority is based on bonus. The insertion and deletion in pririty queue is max of logn
* In Big O terms, O(n\*logn + logn) ~= O(n\*logn)

**Alternate Modelling:**

**Find Node:**  There are various to implement this problem.

**Algorithm:**

|  |
| --- |
| **Input:** File inputPS5.txt contains list of commands / operations.  E.g.  No of use-cases: 2 Deadlines: 1 2 3 1 4 Bonus: 20 40 10 10 20 Deadlines: 4 1 2 2 Bonus: 20 30 10 40  **Output:** File OutputPS5.txt lists the maximum bonus earned and the sequence of Jobs for every test case.  E.g.  90  90  Total number of test cases are 2   For the use case Usecase:1, the maximum bonus earned is 90 For the use case Usecase:2, the maximum bonus earned is 90 For the use case Usecase:1, the tasks were scheduled in Task:1-->Task:2-->Task:5-->Task:3 For the use case Usecase:2, the tasks were scheduled in Task:2-->Task:4-->Task:1  **Function:** The following algorithm gives a high-level overview of the overall program   * **Main**   + Reads the file contents and caches them into uses cases custom datatype   + For each usecase, it performs maximimzation and stores the results   + Finally prints the the results to output file |
|  |