**DSAD Assignment 2**

**PS5 Design Document**

**1. Problem Statement**

You are a part of a class. Your teacher has given a set of questions to the class. Each question takes 1 day to finish the task. Each problem has a deadline, if finished before the deadline you get the extra bonus marks. If you don’t finish the problem before the deadline, you won’t get the bonus. The formulation of the problem is as follows

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | Problem1 | Problem2 | Problem3 | Problem4 | Problem5 |
| **Deadlines** | 1 | 2 | 3 | 1 | 4 |
| **Bonuses** | 20 | 40 | 10 | 10 | 20 |

Maximise the profit by correctly scheduling the problem using greedy-method.

**2. Data Structure Model & Approach to the Problem**

Every problem is represented as a class having three instance variables – Name, Bonus and Deadline.

We read the input and create a list of these problems objects.

Now, we need to use greedy approach to schedule the problems. We do it as follows.

First, we sort the problems in non-increasing order of their bonuses (We are using merge sort algorithm to perform the sorting). We want to be greedy and always consider the problem with highest bonus first and then the next highest and so on.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | Problem2 | Problem1 | Problem5 | Problem3 | Problem4 |
| **Deadlines** | 2 | 1 | 4 | 3 | 1 |
| **Bonuses** | 40 | 20 | 20 | 10 | 10 |

After this, we iterate over this sorted list and schedule the problems in such a way that we don’t miss the deadline but always do it just before that individual problem’s deadline. This means we will only schedule as many problems as the highest unit of time in the *Deadlines* row. Since anything done after deadline is not going to result in any profit, we do not bother to solve those problems even if they remain unsolved.  
  
In the above example, the scheduling is done as follows in steps

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time Slots** | 0-1 | 1-2 | 2-3 | 3-4 |
| **Step1** |  | Problem2 |  |  |
| **Step2** | Problem1 |  |  |  |
| **Step3** |  |  |  | Problem5 |
| **Step4** |  |  | Problem3 |  |

- As Problem2 gives maximum bonus, we first try to schedule that problem. Since it has a deadline of 2 units of time and it only needs 1 unit of time for completion, we schedule it in the time slot of 1-2.

- Next problems with the highest bonus are Problem1 and Problem5. Both have the same bonus. However, problem1 has a deadline of 1 unit of time and Problem5 has deadline of 4 units of time. Hence, we schedule them in first and last time slots respectively.

- Finally, we see that Problem3 and Problem4 both have a same bonus; however, we can no longer schedule Problem4 because we have already scheduled jobs before a deadline of 1 unit of time. But Problem3 has 3 units of time as deadline and time slot 2-3 is vacant, which means we can schedule it to happen in that time interval.

We model the problem in a greedy way as shown above and perform the problem scheduling. The bonus which is accrued is given by the sum of bonuses of the problems which could be scheduled to solve in the respective time slots shown above. For this example,

Bonus = Problem2\_Bonus + Problem1\_Bonus + Problem5\_Bonus + Problem3\_Bonus = 40 + 20 + 20 + 10 = 90

**3. Run time analysis**

Firstly, we need to sort the problem list in non-increasing order of their bonuses. While there are several algorithms to do this, the most efficient algorithms for doing this problem run in O(nlogn) complexity. We have selected merge sort for achieving this objective.  
  
Merge sort divides the problem into halves until there’s only one element in each step and eventually we combine these results to get a sorted sequence of elements. The recurrence relation for such problems can be given by  
  
T(n) = 2T(n/2) + O(n)  
  
We can solve this by master theorem and we obtain the time complexity of this to be nlogn

**4. Alternate way of solving the Problem**