Problem 1: The Launch Day Puzzle – Project Scheduling

FutureTech has secured multiple lucrative contracts, each representing a unique tech project. Every project requires exactly one full day to complete. However, due to limited resources (only one project can be done per day), and looming deadlines from clients, you must schedule these projects wisely.

Each project has:

- A deadline D[i] (latest day by which it must be completed),
- An associated profit P[i].

The CEO has made it clear: "We don't have the bandwidth to do every project. Pick the most profitable combination — as long as each is completed before or on its deadline."

You open your dashboard and see:

Example Input:
Number of Projects: N = 3
Projects: (Deadline, Profit) = (2, 100), (1, 19), (2, 27)

Your Goal: Schedule the projects to maximize total profit, respecting the deadlines.

Expected Output: 127
(*Choose project 1 on day 2 and project 2 on day 1 — project 3 gets skipped.*)

SOLUTION :

To solve this problem, we can use a greedy algorithm that prioritizes the most profitable projects and schedules them as close to their deadlines as possible.

Here's the step-by-step approach:

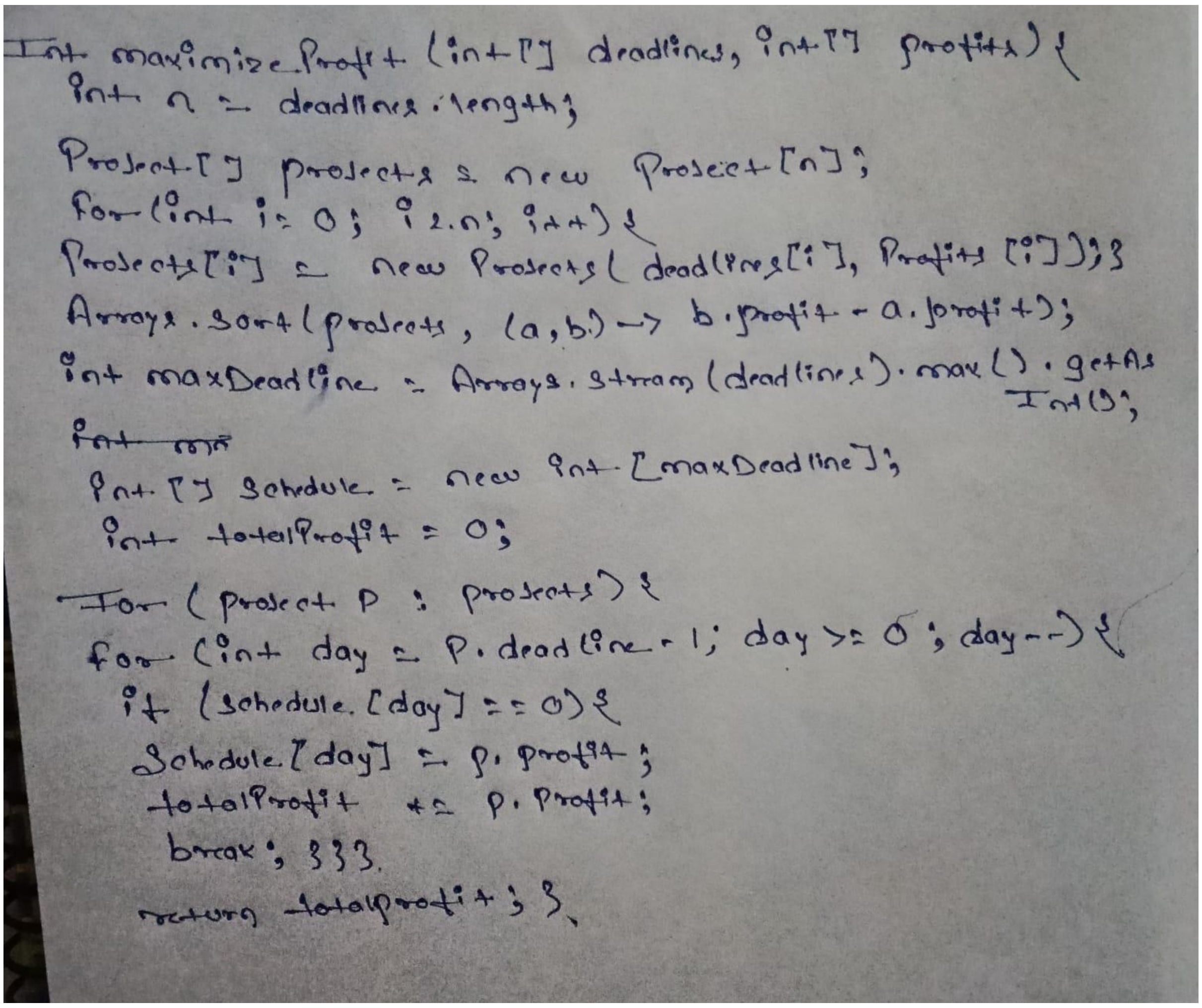
1. Sort by Profit: First, sort all the projects in descending order of their profit. This way, we can always consider the most profitable projects first.

2. Initialize a Schedule: Create an array or list to keep track of which days are occupied. The size of this array should be equal to the maximum deadline among all projects.

3. Schedule Projects: For each project in the sorted list, try to place it on the latest possible day before or on its deadline. If that day is already occupied, move to the previous day until an empty slot is found. If no empty slot is found before the deadline, skip the project.

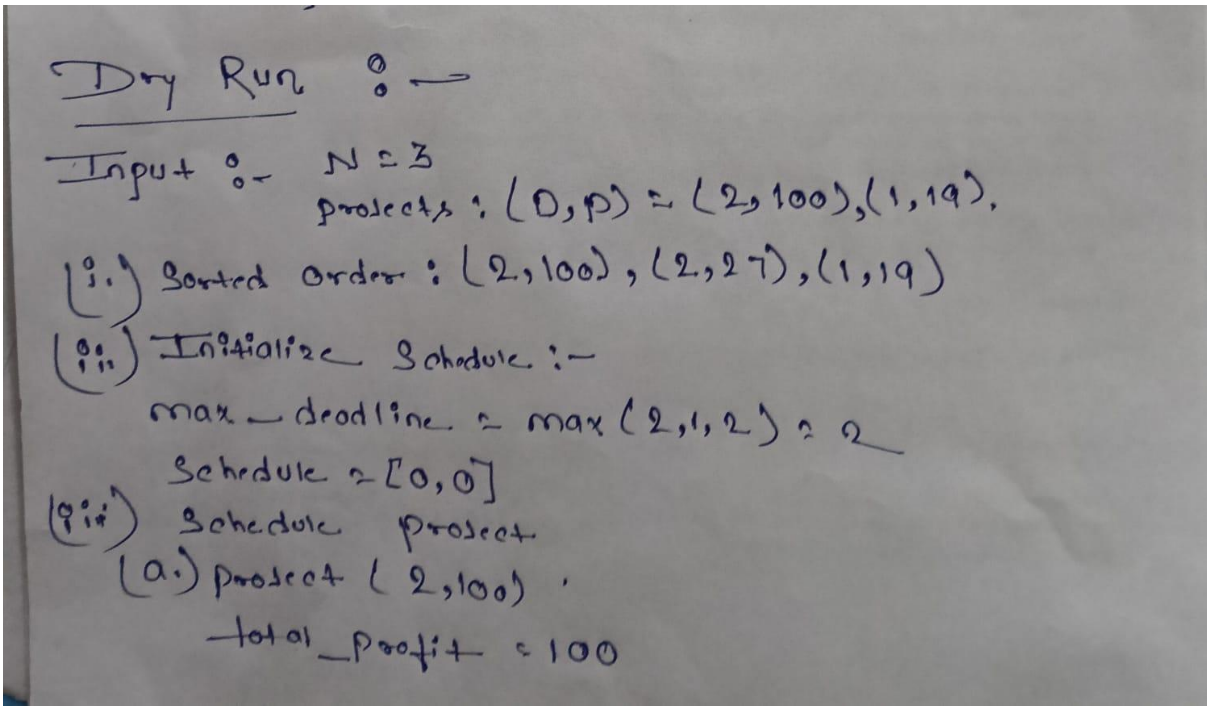
4. Calculate Total Profit: Sum the profits of all scheduled projects.

Pseudocode :-



```
Int maximizeProfit (int[] deadlines, int[] profits) {
  int n = deadlines.length;
  Project[] projects = new Project[n];
  for (int i = 0; i < n; i++) {
  Projects[i] = new Projects( deadlines[i], Profits [i]);}
  Arrays.sort(projects, (a,b) -> b.profit - a.profit);
  int maxDeadline = Arrays.stream (deadlines).max().getAs
                                                   Int();
  ~~int ma~~
   int[] schedule = new int[maxDeadline];
   int totalProfit = 0;
  For ( project P : projects) {
   for (int day = P.deadline - 1; day >= 0 ; day--) {
    if (schedule.[day] == 0) {
     Schedule[day] = p.profit;
     totalProfit += p.Profit;
     break; }}}.
   return totalprofit; }.
```

Dry Run:-



Dry Run :-

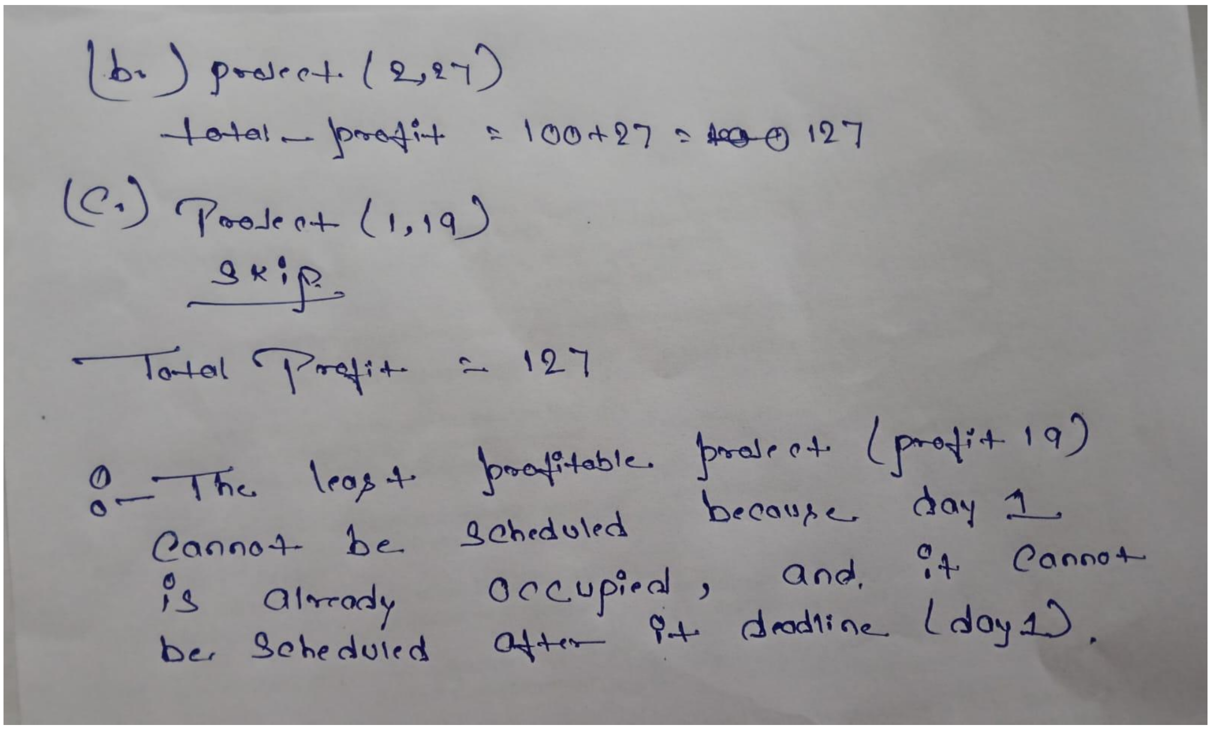
Input :- N = 3
projects : (D, P) = (2, 100), (1, 19).

(i.) Sorted order : (2, 100), (2, 27), (1, 19)

(ii.) Initialize Schedule :-
max_deadline = max(2, 1, 2) = 2
Schedule = [0, 0]

(iii) Schedule project

(a.) project (2, 100).
total_profit = 100



(b.) project (2, 27)
total_profit = 100+27 = ~~100~~ 127

(c.) Project (1, 19)
skip

Total Profit = 127

:- The least profitable project (profit 19) cannot be scheduled because day 1 is already occupied, and it cannot be scheduled after it deadline (day 1).