Optimization mini project:

1. Greedy method:
2. Idea:

* We add all the fields to their first available day for harvesting,which is the first day that has the amount of product plus the amount of product of the field (d(i)) not higher than M.

1. How the heuristic work:

* We first add all the fields into a list, append an additional interger ‘i’ to represent the i field => field = (d, s, e, i)
* Then we sort the list by the value s(i) (the first day a field can be harvested).
* We then iterate through the list, when we encounter field i, we check through all the day from s(i) to e(i), if we found an available day, field i is added to that day and we move on to the next field.
* After we go through the list, we check the harvest plan by removing any day that has the amount of product less than m, along with any fields in that day. Then we got the final harvest plan.
* Pseudo code:

*Function solve(m,M,fields):*

*sort fields by the value start day (s(i))*

*num= the last end day*

*days = [0] \* (num+1)*

*harvested is an empty list*

*for field(d,s,e,i) in fields do:*

*for day in range(s, e+1) do:*

*if days[day] + d <= M then:*

*days[day] += d*

*harvested.append((i, day))*

*break*

*harvested = [(i, day) for i, day in harvested if days[day] >= m]*

*return harvested*

1. Strength and weakness:

* Strengh: the strength of the greedy method is of course its run time.
* Weakness: the greedy method tend to left a lot of fields unharvested when it is harvestable. The most common issue is when a day has the amount of product is smaller than m, instead of moving the fields to the next day and check, it removes all the fields. This also mean if some day lack some amount of product to reach the minimum, they get removed instead of moving some fields from the previous days to it.
* We try to fix this weakness by making Greedy improved.

1. Greedy improved:

* This time, instead of sorting by the end day e(i).
* Then instead of iterating through the fields list, we iterate through the timeline or the day list, from day 1 to the maximum end date e(i).
* In each day, we loop through each field i in the fields list:

+ If the day is between s(i) and e(i) and the day is an available day, we add field i to the day.

+ After we add all possible fields to the day, we check if the amount of product of the day is larger or equal to m. If it is, then all the fields in the day will be added to the harvest plan and we will not loop through these fields again. If not, all the fields are left in the fields list for loop through in the next day.

* After we loop through all the fields, we are left with the harvest plan.
* This method makes sure that if a day is not full, all the fields in it will be check whether they can be in the next day or not. But it also increases the runtime significantly compared to the original greedy method.
* Pseudo code:

*Function solve\_2(m,M,fields):*

*sort fields by the value start day (e(i))*

*num= the last end day*

*days = [0] \* (num+1)*

*harvested is an empty list*

*for day in range(1,num+2) do:*

*for field(d,s,e,i) in fields do:*

*if days[day] + d <= M then:*

*days[day] += d*

*harvested.append((i, day))*

*harvested = [(i, day) for i, day in harvested if days[day] >= m]*

*return harvested*