Maximize Capital (hard)

Problem Statement

Given a set of investment projects with their respective profits, we need to find the **most profitable projects**. We are given an initial capital and are allowed to invest only in a fixed number of projects. Our goal is to choose projects that give us the maximum profit. Write a function that returns the maximum total capital after selecting the most profitable projects.

We can start an investment project only when we have the required capital. Once a project is selected, we can assume that its profit has become our capital.

Example 1:

Input: Project Capitals=[0,1,2], Project Profits=[1,2,3], Initial Capital=1, Number of Projects=2
Output: 6
Explanation:

1. With initial capital of '1', we will start the second project which will give us profit of '2'. Once we selected our first project, our total capital will become 3 (profit + initial capital).
2. With '3' capital, we will select the third project, which will give us '3' profit.

After the completion of the two projects, our total capital will be 6 (1+2+3).

Example 2:

Input: Project Capitals=[0,1,2,3], Project Profits=[1,2,3,5], Initial Capital=0, Number of Projects=3
Output: 8
Explanation:

1. With '0' capital, we can only select the first project, bringing out capital to 1.
2. Next, we will select the second project, which will bring our capital to 3.
3. Next, we will select the fourth project, giving us a profit of 5.

After selecting the three projects, our total capital will be 8 (1+2+5).

Try it yourself

Try solving this question here:

Java | Python3 | JS | C++

```
const find_maximum_capital = function(capital, profits, numberOfProjects, initialCapital) {
  //TODO: Write your code here
  return -1;
};


console.log(`Maximum capital: ${find_maximum_capital([0, 1, 2], [1, 2, 3], 2, 1)}`)
console.log(`Maximum capital: ${find_maximum_capital([0, 1, 2, 3], [1, 2, 3, 5], 3, 0)}`)

```

RUN SAVE RESET

Solution

While selecting projects we have two constraints:

1. We can select a project only when we have the required capital.
2. There is a maximum limit on how many projects we can select.

Since we don't have any constraint on time, we should choose a project, among the projects for which we have enough capital, which gives us a maximum profit. Following this greedy approach will give us the best solution.

While selecting a project, we will do two things:

1. Find all the projects that we can choose with the available capital.
2. From the list of projects in the 1st step, choose the project that gives us a maximum profit.

We can follow the **Two Heaps** approach similar to Find the Median of a Number Stream. Here are the steps of our algorithm:

1. Add all project capitals to a min-heap, so that we can select a project with the smallest capital requirement.
2. Go through the top projects of the min-heap and filter the projects that can be completed within our available capital. Insert the profits of all these projects into a max-heap, so that we can choose a project with the maximum profit.
3. Finally, select the top project of the max-heap for investment.
4. Repeat the 2nd and 3rd steps for the required number of projects.

Code

Here is what our algorithm will look like:

Java | Python3 | C++ | JS

```
    minCapitalHeap.push([capital[i], i]);
  }

  // let's try to find a total of 'numberOfProjects' best projects
  let availableCapital = initialCapital;
  for (i = 0; i < numberOfProjects; i++) {
    // find all projects that can be selected within the available capital and insert them in a max-heap
    while (minCapitalHeap.length > 0 && minCapitalHeap.peek()[0] <= availableCapital) {
      const [capital, index] = minCapitalHeap.pop();
      maxProfitHeap.push([profits[index], index]);
    }

    // terminate if we are not able to find any project that can be completed within the available capita
    if (maxProfitHeap.length === 0) {
      break;
    }

    // select the project with the maximum profit
    availableCapital += maxProfitHeap.pop()[0];
  }

  return availableCapital;
}


console.log(`Maximum capital: ${find_maximum_capital([0, 1, 2], [1, 2, 3], 2, 1)}`);
console.log(`Maximum capital: ${find_maximum_capital([0, 1, 2, 3], [1, 2, 3, 5], 3, 0)}`);
```

RUN | SAVE | RESET

Close

Output 8.232s

```
Maximum capital: 6
Maximum capital: 8
```

Time complexity

Since, at the most, all the projects will be pushed to both the heaps once, the time complexity of our algorithm is $O(NlogN + KlogN)$, where 'N' is the total number of projects and 'K' is the number of projects we are selecting.

Space complexity

The space complexity will be $O(N)$ because we will be storing all the projects in the heaps.