

5736. Single-Threaded CPU

My Submissions (/contest/weekly-contest-237/problems/single-threaded-cpu/submissions/)

Back to Contest (/contest/weekly-contest-237/)

You are given n tasks labeled from 0 to $n - 1$ represented by a 2D integer array `tasks`, where `tasks[i] = [enqueueTimei, processingTimei]` means that the i^{th} task will be available to process at `enqueueTimei` and will take `processingTimei` to finish processing.

You have a single-threaded CPU that can process **at most one** task at a time and will act in the following way:

- If the CPU is idle and there are no available tasks to process, the CPU remains idle.
- If the CPU is idle and there are available tasks, the CPU will choose the one with the **shortest processing time**. If multiple tasks have the same shortest processing time, it will choose the task with the smallest index.
- Once a task is started, the CPU will **process the entire task** without stopping.
- The CPU can finish a task then start a new one instantly.

User Accepted:	0
User Tried:	0
Total Accepted:	0
Total Submissions:	0
Difficulty:	Medium

Return the order in which the CPU will process the tasks.

Example 1:

Input: `tasks = [[1,2],[2,4],[3,2],[4,1]]`
Output: `[0,2,3,1]`
Explanation: The events go as follows:
- At time = 1, task 0 is available to process. Available tasks = {0}.
- Also at time = 1, the idle CPU starts processing task 0. Available tasks = {}.
- At time = 2, task 1 is available to process. Available tasks = {1}.
- At time = 3, task 2 is available to process. Available tasks = {1, 2}.
- Also at time = 3, the CPU finishes task 0 and starts processing task 2 as it is the shortest. Available tasks = {1}.
- At time = 4, task 3 is available to process. Available tasks = {1, 3}.
- At time = 5, the CPU finishes task 2 and starts processing task 3 as it is the shortest. Available tasks = {1}.
- At time = 6, the CPU finishes task 3 and starts processing task 1. Available tasks = {}.
- At time = 10, the CPU finishes task 1 and becomes idle.

Example 2:

Input: `tasks = [[7,10],[7,12],[7,5],[7,4],[7,2]]`
Output: `[4,3,2,0,1]`
Explanation: The events go as follows:
- At time = 7, all the tasks become available. Available tasks = {0,1,2,3,4}.
- Also at time = 7, the idle CPU starts processing task 4. Available tasks = {0,1,2,3}.
- At time = 9, the CPU finishes task 4 and starts processing task 3. Available tasks = {0,1,2}.
- At time = 13, the CPU finishes task 3 and starts processing task 2. Available tasks = {0,1}.
- At time = 18, the CPU finishes task 2 and starts processing task 0. Available tasks = {1}.
- At time = 28, the CPU finishes task 0 and starts processing task 1. Available tasks = {}.
- At time = 40, the CPU finishes task 1 and becomes idle.

Constraints:

- `tasks.length == n`
- `1 <= n <= 105`
- `1 <= enqueueTimei, processingTimei <= 109`

JavaScript

```
1 /**
2  * @param {number[][]} tasks
3  * @return {number[]}
4  */
5 var getOrder = function(tasks) {
6
7 };
```