【华为OD机考 统一考试机试C卷】GPU 调度/执行时长（C++ Java JavaScript Python C语言）

华为OD机考:统一考试 C卷 + D卷 + B卷 +A卷

目前在考C卷，经过两个月的收集整理，C卷真题已基本整理完毕
抽到原题的概率为2/3到3/3，也就是最少抽到两道原题。请注意：大家刷完C卷真题，最好要把B卷的真题刷一下，因为C卷的部分真题来自B卷。
另外订阅专栏还可以联系笔者开通在线 OJ 进行刷题，提高刷题效率。
真题目录：华为OD机考机试 真题目录（C卷 + D卷 + B卷 + A卷） + 考点说明
专栏：2023华为OD机试(B卷+C卷+D卷)（C++JavaJSPy）
华为OD面试真题精选：华为OD面试真题精选
在线OJ：点击立即刷题，模拟真实机考环境

题目描述

为了充分发挥GPU[算力]，需要尽可能多的将任务交给GPU执行，现在有一个任务数组，数组元素表示在这1秒内新增的任务个数且每秒都有新增任务。

假设GPU最多一次执行n个任务，一次执行耗时1秒，在保证GPU不空闲情况下，最少需要多长时间执行完成。

输入描述

- 第一个参数为GPU一次最多执行的任务个数，取值范围[1, 10000]
- 第二个参数为任务数组长度，取值范围[1, 10000]
- 第三个参数为任务数组，数字范围[1, 10000]

输出描述

执行完所有任务最少需要多少秒。

用例

输入	3 5 1 2 3 4 5
输出	6
说明	一次最多执行3个任务，最少耗时6s

输入	4 5 5 4 1 1 1
输出	5
说明	一次最多执行4个任务，最少耗时5s

C++

```cpp
#include <iostream>
using namespace std;

int main() {
    int maxTasks;
    cin >> maxTasks;

    int taskArrLen;
    cin >> taskArrLen;

    int* taskArr = new int[taskArrLen];
    for (int i = 0; i < taskArrLen; i++) {
        cin >> taskArr[i];
    }

    int currentTasks = 0;
    int time = 0;
    int index = 0;

    while (currentTasks != 0 || index != taskArrLen) {
        if (index < taskArrLen) {

```

```
            currentTasks += taskArr[index];
            index++;
        }

        currentTasks -= maxTasks;

        if (currentTasks < 0) currentTasks = 0;

        time++;
    }

    cout << time << endl;

    delete[] taskArr;

    return 0;
}
```

javascript

```
const readline = require('readline');

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});

let maxTasks = 0;
let taskArrLen = 0;
let taskArr = [];
let currentTasks = 0;
let time = 0;
let index = 0;

rl.on('line', (line) => {
  if (maxTasks === 0) {
    maxTasks = parseInt(line.trim());
  } else if (taskArrLen === 0) {
    taskArrLen = parseInt(line.trim());
```

```
  } else {

      taskArr = line.split(" ").slice(0, taskArrLen).map((ele) => parseInt(ele));

  }
});

rl.on('close', () => {
   while (currentTasks !== 0 || index !== taskArr.length) {
    if (index < taskArr.length) {
      currentTasks += taskArr[index];
      index++;
    }
    currentTasks -= maxTasks;
    if (currentTasks < 0) currentTasks = 0;
    time++;
  }

  console.log(time);
});
```

java

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);

        // 输入GPU一次最多执行的任务个数
        int maxTasks = in.nextInt();

        // 输入任务数组长度
        int taskArrLen = in.nextInt();

        // 输入任务数组
        int[] taskArr = new int[taskArrLen];
        for (int i = 0; i < taskArrLen; i++) {
```

```
            taskArr[i] = in.nextInt();
        }

        // 当前任务个数
        int currentTasks = 0;

        // 执行任务所需时间
        int time = 0;

        // 任务数组索引
        int index = 0;

        // 当前任务个数不为0或任务数组未遍历完时继续执行
        while (currentTasks != 0 || index != taskArr.length) {
            // 如果任务数组还有任务未遍历
            if (index < taskArr.length) {
                // 将新增的任务加到当前任务个数中
                currentTasks += taskArr[index];
                index++;
            }

            // 执行一次最多的任务个数
            currentTasks -= maxTasks;

            // 如果当前任务个数小于0，将其置为0
            if (currentTasks < 0) currentTasks = 0;

            // 执行任务所需时间加1
            time++;
        }

        // 输出执行完所有任务所需的最少时间
        System.out.println(time);
    }
}
```

python

```
maxTasks = int(input())
taskArrLen = int(input())
```

```python
taskArr =  list(map(int, input().split()))


currentTasks = 0
time = 0
index = 0

while currentTasks != 0 or index != len(taskArr):
    if index < len(taskArr):
        currentTasks += taskArr[index]
        index += 1

    currentTasks -= maxTasks
    if currentTasks < 0:
        currentTasks = 0

    time += 1

print(time)
```

C语言

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int maxTasks;
    scanf("%d", &maxTasks);

    int taskArrLen;
    scanf("%d", &taskArrLen);

    // 使用malloc分配动态数组
    int* taskArr = (int*)malloc(taskArrLen * sizeof(int));
    for (int i = 0; i < taskArrLen; i++) {
        scanf("%d", &taskArr[i]);
    }

    int currentTasks = 0;
    int time = 0;
```

```
    int index = 0;

    // 当前任务不为0或者还有任务未添加到currentTasks时，继续执行
    while (currentTasks != 0 || index != taskArrLen) {
        if (index < taskArrLen) {
            // 将新增的任务添加到currentTasks
            currentTasks += taskArr[index];
            index++;
        }

        // 每秒执行maxTasks个任务
        currentTasks -= maxTasks;

        // 如果执行后任务数为负，则置为0
        if (currentTasks < 0) currentTasks = 0;

        // 时间增加
        time++;
    }

    printf("%d\n", time);

    // 释放动态分配的内存
    free(taskArr);

    return 0;
}
```

完整用例

用例1

```
3
5
1 2 3 4 5
```

用例2

```
4
5
5 4 1 1 1
```

用例3

```
2
3
1 1 1
```

用例4

```
1
5
1 1 1 1 1
```

用例5

```
5
4
1 2 3 4
```

用例6

```
10
10
1 2 3 4 5 6 7 8 9 10
```

用例7

```
7
6
3 4 5 6 7 8
```

用例8

```
100
10
```

```
10 20 30 40 50 60 70 80 90 100
```

用例9

```
5
8
1 1 1 1 1 1 1 1
```

用例10

```
3
10
2 2 2 2 2 2 2 2 2 2
```

文章目录

用例9
用例10