**《软件技术进展》大作业**

软件工程 院/系 软件工程 专业 20 级 1 班

实验时间： 2022 年 7 月 2 日

姓 名： 李阳姿 学 号： 20331019

实验名称 ： 同时运行 N 台电脑的最长时间

**目录**

[一、题目 2](#_Toc31664)

[二、 Github仓库地址 3](#_Toc3135)

[三、 解题过程描述 3](#_Toc28703)

[四、 源代码 3](#_Toc7944)

[1、 引入库 3](#_Toc28962)

[2、 思路 3](#_Toc21251)

[3、 排序算法 3](#_Toc1730)

[4、 消耗电池电量 4](#_Toc31121)

[5、 计算 4](#_Toc26732)

[6、 发现思路有误，更换思路 5](#_Toc12727)

[7、代码总结 5](#_Toc4405)

**一、题目**

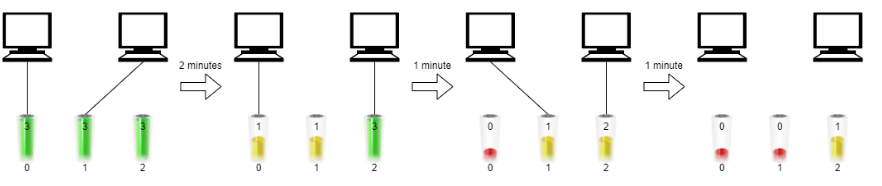
你有 n 台电脑。给你整数 n 和一个下标从 0 开始的整数数组 batteries ，其中第 i 个电池可以让一台电脑 运行 batteries[i] 分钟。你想使用这些电池让 全部 n 台电脑 同时 运行。

一开始，你可以给每台电脑连接 至多一个电池 。然后在任意整数时刻，你都可以将一台电脑与它的电池断开连接，并连接另一个电池，你可以进行这个操作 任意次 。新连接的电池可以是一个全新的电池，也可以是别的电脑用过的电池。断开连接和连接新的电池不会花费任何时间。

注意，你不能给电池充电。

请你返回你可以让 n 台电脑同时运行的 最长 分钟数。

示例 1：



输入：n = 2, batteries = [3,3,3]

输出：4

解释：

一开始，将第一台电脑与电池 0 连接，第二台电脑与电池 1 连接。

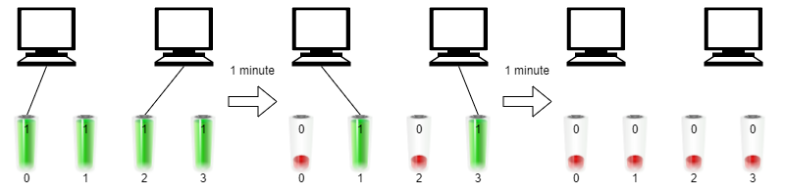
2 分钟后，将第二台电脑与电池 1 断开连接，并连接电池 2 。注意，电池 0 还可以供电 1 分钟。

在第 3 分钟结尾，你需要将第一台电脑与电池 0 断开连接，然后连接电池 1 。

在第 4 分钟结尾，电池 1 也被耗尽，第一台电脑无法继续运行。

我们最多能同时让两台电脑同时运行 4 分钟，所以我们返回 4 。

示例 2：



输入：n = 2, batteries = [1,1,1,1]

输出：2

解释：

一开始，将第一台电脑与电池 0 连接，第二台电脑与电池 2 连接。

一分钟后，电池 0 和电池 2 同时耗尽，所以你需要将它们断开连接，并将电池 1 和第一台电脑连接，电池 3 和第二台电脑连接。

1 分钟后，电池 1 和电池 3 也耗尽了，所以两台电脑都无法继续运行。

我们最多能让两台电脑同时运行 2 分钟，所以我们返回 2 。

1. **Github仓库地址**

|  |
| --- |
| https://github.com/20220303/sse\_0702\_bh.git |

1. **解题过程描述**

|  |
| --- |
| 先将数组排序，然后假设有N台电脑，有M块电池  1.N>M,则为0 2.将数组排序 3.按顺序从M数组的最 大 的N个提出来【这里变成最大的N个】 4.N个中每一个减去其中最小的元素，总的使用时间加上这个最小元素 5.直到M中没有N个有电的电池 6.最后的到N台电脑的最长使用时间 |

1. **源代码**
2. **引入库**

|  |
| --- |
| import numpy as np |

1. **思路**

|  |
| --- |
| 先将数组排序，然后假设有N台电脑，有M块电池  1.N>M,则为0  2.将数组排序  3.按顺序从M数组的最小的N个提出来  4.N个中每一个减去其中最小的元素，总的使用时间加上这个最小元素  5.N在M中右移直到最小位不是零，【右移一位也行，就是慢一点】  6.如果N最右边已经到达了最大，那么就减去最小的元素值就结束了  7.最后的到N台电脑的最长使用时间 |

1. **排序算法**

|  |
| --- |
| #先采用冒泡排序,采用start，end的部分排序  #修改冒泡排序中的功能,使其能够消耗电池中的电量  def bubbleSort(nums):  for i in range(start,len(nums)):  for j in range(start,len(nums)-i-1):  if nums[j] > nums[j+1]:  nums[j], nums[j+1] = nums[j+1], nums[j]  return nums |

1. **消耗电池电量**

|  |
| --- |
| def use\_battery(nums,start,end):  min\_battery = nums[start]  for i in range(start,end):  nums[i] = nums[i] - min\_battery  return min\_battery |

1. **计算**

|  |
| --- |
| #设计算法  N = 3  M = [6,8,3,2,9,1,0]  # M = [2,3,4,1,5,6]  start = 0  print("根据电量排序好的电池：",bubbleSort(M))  sum\_use = 0  while((start+N) <= len(M)):  sum\_use += use\_battery(M,start,start+N)  print("更换电源：",bubbleSort(M))  start += 1    print("同时运行",N ,"台电脑的最长时间:",sum\_use) |
| ANS:  根据电量排序好的电池： [0, 1, 2, 3, 6, 8, 9]  更换电源： [0, 1, 2, 3, 6, 8, 9]  更换电源： [0, 0, 1, 2, 6, 8, 9]  更换电源： [0, 0, 0, 1, 5, 8, 9]  更换电源： [0, 0, 0, 0, 4, 7, 9]  更换电源： [0, 0, 0, 0, 0, 3, 5]  同时运行 3 台电脑的最长时间: 7 |

1. **发现思路有误，更换思路**

|  |
| --- |
| 先将数组排序，然后假设有N台电脑，有M块电池  1.N>M,则为0 2.将数组排序 3.按顺序从M数组的最 大 的N个提出来【这里变成最大的N个】 4.N个中每一个减去其中最小的元素，总的使用时间加上这个最小元素 5.直到M中没有N个有电的电池 6.最后的到N台电脑的最长使用时间 |
| N = 4  M = [6,8,3,2,9,1,0]  # M = [2,3,4,1,5,6]  # N>M的情况  # M = [6,8]  start = 0  print("根据电量排序好的电池：",bubbleSort(M))  sum\_use = 0  while(M[len(M)-N] and len(M)>N):  sum\_use += use\_battery(M,len(M)-N,len(M))  print("更换电源：",bubbleSort(M))  print("同时运行",N ,"台电脑的最长时间:",sum\_use) |
| ANS:  根据电量排序好的电池： [0, 1, 2, 3, 6, 8, 9]  更换电源： [0, 0, 1, 2, 3, 5, 6]  更换电源： [0, 0, 0, 1, 1, 3, 4]  更换电源： [0, 0, 0, 0, 0, 2, 3]  同时运行 4 台电脑的最长时间: 6  ANS：   1. M的情况   根据电量排序好的电池： [6, 8]  同时运行 4 台电脑的最长时间: 0 |

**7、代码总结**

|  |
| --- |
| import numpy as np  def bubbleSort(nums):  for i in range(start,len(nums)):  for j in range(start,len(nums)-i-1):  if nums[j] > nums[j+1]:  nums[j], nums[j+1] = nums[j+1], nums[j]  return nums  def use\_battery(nums,start,end):  min\_battery = nums[start]  for i in range(start,end):  nums[i] = nums[i] - min\_battery  return min\_battery  N = 4  M = [6,8,3,2,9,1,0]  # M = [2,3,4,1,5,6]  # N>M的情况  # M = [6,8]  start = 0  print("根据电量排序好的电池：",bubbleSort(M))  sum\_use = 0  while(M[len(M)-N] and len(M)>N):  sum\_use += use\_battery(M,len(M)-N,len(M))  print("更换电源：",bubbleSort(M))  print("同时运行",N ,"台电脑的最长时间:",sum\_use) |