Sereja and Order 解题报告

石家庄市第二中学 张若天

1. 题目来源

Codechef Nov 14

2. 题目大意

有 N 个任务,每个任务都要在两台机器上运行,第 i 个任务需要在第一台机器上运行 A[i]秒,在第二台机器上运行 B[i]秒。一台机器不能同时运行两个任务,一个任务也不能同时在两台机器上运行。求完成所有任务所需的最少时间。

数据范围: $1 \le A[i], B[i] \le 10^5, 1 \le N \le 10^4$ 。

3. 算法讨论

3.1. 答案的下界

一个任务不能在两台机器同时运行,就意味着答案不能小于每个任务的 A[i]+B[i]。

每个任务都需要在每台电脑上运行,就意味着答案不能小于 $\max(\sum Ai, \sum Bi)$ 。

所以所需时间的下界是 $\max(\max_i(A[i] + B[i]), \max(\sum Ai, \sum Bi))$ 。

3.2. 下界是可达的

现在我们要证明所有的情况下下界都是可以达到的。

3.3. 第一种情况: $max(A[i] + B[i]) \ge max(\sum Ai, \sum Bi)$

我们设取到 $\max(A[i] + B[i])$ 的任务为任务 p。

所以 $A[p] + B[p] \ge A[1] + A[2] + \cdots + A[p] + \cdots + A[n]$ 。

以及 $A[p] + B[p] \ge B[1] + B[2] + \dots + B[p] + \dots + B[n]$ 。

令第一个式子两边同时减去A[p],第二个式子两边同时减去B[p]。

得到 $B[p] \ge \sum Ai - A[p]$ 和 $A[p] \ge \sum Bi - A[p]$ 。

这样就可以安排第 p 个任务从 0 时开始做 A。在[0,A[p]]这段时间里,其他任务可以做完 B。然后 p 任务在[A[p],A[p]+B[p]]这段时间内做 B,其他任务又可以做完 A。

这种情况解决。

3.4. 第二种情况: $max(\sum Ai, \sum Bi) > max(A[i] + B[i])$

在下面内容中,设 $t = \max(\sum Ai, \sum Bi)$ 。

3.4.1. 尝试贪心

一个比较基本简单的贪心是这样的:尝试按照任意的一个顺序 在 A 机器上运行,再按照相反的顺序在 B 上运行。

对于每个任务,如果它在 A 上运行的较早,就会在 B 上运行的较晚。

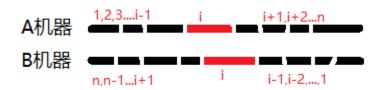
形式化的: 对于第 i 个任务在 A 机器 $[\sum_{j < i} A_j, \sum_{j \le i} A_j]$ 运行,在 B 机器 $[t - \sum_{j \le i} B_j, t - \sum_{j < i} B_j]$ 运行。

3.4.2. 贪心会使任务相交

虽然一个任务在 A 上运行较早就会在 B 上运行较晚,但在中间部分还是可能产生冲突。即一个任务在 A 机器的运行时间段与在 B 机器的相交。这是不允许的。

3.4.3.通过移动来避免相交

显然最多只会有一个任务相交,设这个任务编号为i。



上面是一个相交的情况,其他相交情况可以通过交换 A, B 或对称来转化到上面的情况。

我们考虑通过移动任务 i 来避免这次相交,又不引入其他的相交情况。

具体来说,我们可以把上图的 i 任务在 B 机器移至最后,然后 把 i-1, i-2, …, 1 这些任务同一提前。

观察到若 i 任务移动后不相交, i-1, i-2, ···, 1 这些任务也不会相交。

我们需要一个初始的顺序,来保证出现相交情况后移动一次不相交。

若初始顺序按照A[i] - B[i] 从小到大排序,可以保证 $A[1] + A[2] + \cdots + A[i] \le t - B[i]$,就能达到上面要求。 时间复杂度: O(nlog(n)),空间复杂度: O(n) 。

3.5. 随机化算法

每次随机一个顺序判断是否合法。

因为在上面算法中 1[~]i-1 与 i+1[~]n 的相对顺序不对是否合法产生影响,所以可行解比较多,可以使用随机化算法。

4. 考察内容

排序, 贪心, 逻辑分析