Algorithm Design and Analysis: Assignment 3

钟赟 202028013229148

2020 年 11 月 25 日

1 Monkeys and Bananas

Solution

1.1 Algorithm Description

首先把猴子和香蕉的位置按照从左到右的顺序编号，也即坐标从小到大。令每个猴子选择其顺序对应位置的香蕉，比如从左数第一只猴子拿从左数第一根香蕉、从左数第二只猴子拿从左数第二根香蕉等。

伪代码见 Algorithm 1。

Algorithm 1 MONKEYS_AND_BANANAS algorithm

```
function MONKEYS_AND_BANANAS(monkeys, bananas)
    Sort monkeys in ascending order;
    Sort bananas in ascending order;
    for i = 0 to bananas.length do
        time = max (time, |bananas[i] − monkeys[i]|);
    end for
    return time ;
end function
```

1.2 Greedy-choice Property

Greedy 选择策略为：每个猴子选择其对应位置的香蕉，例如：左起第一个猴子应该拿左起的第一根香蕉。

最优子结构：第 i 只猴子拿第 i 根香蕉。

1.3 Correctness Proof

假设根据 1.2 中的 Greedy 选择策略，所得到的时间不是最小的。

根据假设，存在 i，第 i 个猴子没有拿到第 i 根香蕉，设其拿了第 j 根香蕉。则第 j 只猴子拿了第 k 根香蕉...，以此类推。这些乱序对构成了多个环。不失一般性，我们假设前 i 个猴子和前 i 根香蕉

乱序，构成长度为 i 的环，且这个环不能拆分成更小的环。我们只需证明，对长度为 i 的环，其花费的时间大于按照顺序拿即可。

第 1 个猴子不拿第1根香蕉，则必定存在另一根香蕉比第1根香蕉离它更近；此时对于其他所有的猴子来说，都不愿意跑更远的路去拿第一根香蕉。经过多步 Greedy 决策后，第 i 只猴子只能去拿第一根香蕉，因为长度为 i 的环，且这个环不能拆分成更小的环。我们只需证明，对长度为第 i 只猴子拿第1根香蕉的时间大于它们按顺序拿香蕉所需要的时间，最终的结果取最大值，也将大于它们按顺序拿香蕉的时间。

综上，乱序对中的猴子拿香蕉所需要的时间大于它们按顺序拿香蕉的时间，假设不成立，算法正确性得证。

1.4 Complexity

排序的复杂度为 $\mathcal{O}(nlogn)$，数组遍历的复杂度为$\mathcal{O}(n)$，故时间复杂度为$\mathcal{O}(nlogn)$。

2 Job Schedule

Solution

2.1 Algorithm Description

算法描述：由于 PCs 数量无限制，则作业调度在 PCs 上耗时排序与在 supercomputer 上的耗时排序无关。因为不论按照什么顺序，supercomputer 总是在忙碌中，没有空闲，不影响总耗时的计算。因此我们考虑在 PCs 上的调度：每次决策时选择剩下 job 中，f_i 最大的 job 来运行。遍历每个 job，每个 job 运行的最少的时间为：之前所有 job 在 PCs 上累积剩余的时间、上一个 job 在 PCs 上剩余的时间、以及当前job的 f_i ，三者取最大值。

伪代码见 Algorithm 2。

Algorithm 2 JOB_SCHEDULE algorithm

function JOB_SCHEDULE(p, f)
  Sort p and f in descending order of f.
  for $i = 0$ to $p.length$ **do**
    $minTime += p[i]$;
    if $i == 0$ **then**
      $remainTime = f[0]$;
    else
      $remainTime = \max\ (remainTime - p[i], \max(f[i-1] - p[i], f[i]))$;
    end if
  end for
  return $minTime + remainTime$;
end function

2.2 Greedy-choice Property

根据 Greedy 思想，我们希望 supercomputer 和 PCs 尽量没有空闲。因此，我们选择先运行 f_i 大的 job ，这样在 supercomputer 运行的时候，尽量让 PCs 不空闲。

最优子结构：每次决策选择剩下 job 中，f_i 最大的 job。

2.3 Correctness Proof

假设存在一个最佳 job schedule 序列 $J_0, J_1, \ldots, J_n$ ，其中对于 job J_i 和 J_{i+1}，$f_i < f_{i+1}$。

设 pte 为在运行到某个 job 之前，在 supercomputer 上运行的总时间， fte 为当前运行的总时间。当对 job J_i 和 J_{i+1} 做决策时，有

$pte+ = p[i];$

$fte_1 = max(fte, pte + f[i]);$

$pte+ = p[i+1];$

$fte_2 = max(fte_1, pte + f[i+1]);$

如果我们交换job J_i 和 J_{i+1}的运行顺序，则有

$pte+ = p[i+1];$

$fte = max(fte_1, pte + f[i+1]);$

$pte+ = p[i];$

$fte_2 = max(fte_1, pte + f[i]);$

改变运行顺序对 pte 没有影响。由于 $f[i] < f[i+1]$ ，那么在第一种运行顺序中有 $fte_2 > fte_1$，在第二种运行顺序中有 $fte_2 \geq fte_1$。因此，交换J_i 和 J_{i+1}的运行顺序不会增大fte。

综上，贪心算法是正确的。

2.4 Complexity

排序的复杂度为$\mathcal{O}(nlogn)$，遍历数组的复杂度为$\mathcal{O}(n)$，因此总复杂度为$\mathcal{O}(nlogn)$。

3 Cross the River

Solution

3.1 Algorithm Description

首先将所有人的体重从小到大排序。优先运送剩下的人中体重最重的人，如果他能够跟体重最轻的人一起乘船，则一起乘船，如果不能，他就单独乘船。

伪代码见 Algorithm 3 。

3.2 Greedy-choice Property

根据 Greedy 的思想，我们想要船的个数最少，最少的情况是乘船人数的 1/2 ，也就是说尽量让两个人乘坐一条船。为了最大化利用船的载量，我们应该优先让体重最大的乘客上船，并且尽量让其跟

Algorithm 3 CROSS_THE_RIVER algorithm

function CROSS_THE_RIVER($num, limit, weights$)
  Sort $weights$ in ascending order.
  $numBoat = 0$;
  for $(i = 0, j = num - 1; i \leq j; j--)$ **do**
    $sum = weights[i] + weight[j]$;
    if $sum < limit$ **then**
      $i++$;
    end if
    $numBoat++$;
  end for
  return $numBoat$;
end function

别人一起乘船，因此考虑体重最轻的乘客，如果他不能跟体重最大的乘客一起上船，那么就没有乘客能够跟体重最大的乘客上船，他只能自己上船。因此可以保证尽量让两人一起乘船。

最优子结构：每次决策选择剩下的乘客中体重最大的乘客优先上船，在考虑体重最轻的乘客能否跟他一起上船。

3.3 Correctness Proof

假设根据 3.1 中的 Greedy 算法，得到的船的数量不是最少的。

根据假设知，存在两位乘客i和j，$i < j$，他们的体重之和 $weights[i]+weight[j] \leq limit$，却分别单独乘坐了两条船。由于数组有序，得知 $weights[i] \leq weights[j]$。当对乘客 j 进行决策时，$weights[j] < limits$ 且 $weights[i]+weight[j] \leq limit$，那么如果乘客$i$前的乘客都已经坐船走了，乘客 j 会选择跟乘客 i 一起乘船走；如果乘客 i 前还有别的乘客，那么乘客 j 会选择跟当前体重最轻的乘客一起乘船走。因此，乘客 j 不可能单独乘船走，与假设矛盾。

综上，假设不成立，该贪心算法是正确的。

3.4 Complexity

排序的复杂度为$\mathcal{O}(nlogn)$，遍历数组的复杂度为$\mathcal{O}(n)$，因此总复杂度为$\mathcal{O}(nlogn)$。