中国科学院大学

University of Chinese Academy of Sciences

计算机算法设计与分析
第三次作业

201818013229056 苗天昊

2018 年 11 月 22 日

1 Greedy Algorithm

Given a list of n natural numbers $d_1, d_2, ..., d_n$, show how to decide in polynomial time whether there exists an undirected graph $G = (V, E)$ whose node degrees are precisely the numbers $d_1, d_2, ..., d_n$. G should not contain multiple edges between the same pair of nodes, or "loop" edges with both endpoints equal to the same node.

问题分析与证明：

先考虑最小情况。

1）当图 G 没有顶点时，不存在顶点的度。当顶点度的集合为空时，可以构成一个空图。

2）当图 G 只有一个顶点时，顶点度的集合非空，并且只有一个元素，其值为 0 时可以构成一个顶点的图。

3）当图 G 有两个顶点时，顶点度的集合为 $\{d_1, d_2\}$。不妨假设 d_1 是较大值，如果 d_1 大于 2-1=1，则肯定无法构成无向图。并且 d_1 值非负，只要顶点的度有一个值为负数，则该顶点度数集肯定不能构成无向图。所以 d_1 的取值只有两种可能 0 和 1，如果 $d_1 = 1$，所以顶点之间有一条边，将度数为 d_1 的顶点 V_1 删除，则 d_2 对应的顶点 V_2 少了一条与 V_1 相连的边。其子问题转化为从 $\{d_2 - 1\}$ 的顶点度数集中判断其能否组成无向图。d_2 的值只有当且仅当 1 的时候可以在 $d_1 = 1$ 的情况下构成无向图。如果 $d_1 = 0$，则其子问题转化为从 $\{d_2\}$ 的顶点度数集中判断其能否组成无向图，当切仅当 d_2 的值为 0 时，可以构成无向图。

现考虑一般情况，给出有 n 个顶点的顶点度数集合 $\{d_1, d_2, ..., d_n\}$，同样，我们将该顶点度数集进行排序，不妨假设 $d_1, d_2, ..., d_n$ 是经过排序的，并且 $d_1 >= d_2 >= ... >= d_n$，令 d_1 的值

为 k，即 d_1 对应的顶点 V_1 有 k 条边，现将顶点 V_1 删除，有边连接的各顶点度数减 1，其子问题就是从 $d_2-1, d_3-1, ., d_k-1, d_{k+1}.., d_n$ 中判断其能否构成无向图。

需要证明，当前度数最大 (其值为 k) 的点为 V_1，其一定与度数序列中的后 k 个顶点有边。假设存在一个顶点 $V_i(2 \le i \le k)$，其与 V_1 不存在边连接。因为 V_1 的度数为 k，所以一定存在个值 $j > k$，V_j 与 V_1 存在边。由于 $d_i > d_k > d_j$，所以一定存在一个顶点 V_k 与 V_i 存在边，而与 V_j 不存在边，即一定存在边 (V_i, V_k)。于是，可以将边 (V_1, V_j) 和 $V_i, V_k)$ 删除，添加边 (V_1, V_i) 和 (V_j, V_k)，变化后顶点度数集合不变，可以依次调整边，使度数为 d_1 的点与 $d_2, d_3, ..., d_k$ 存在边。

Greedy 1 Greedy 1

```
sort d_1, d_2, ..., d_n as d[n] //降序排列 d_1, d_2, ..., d_n
if d[0] < 0 || d[0] > n-1 then
    return False
end if
for i = 0 to n − 1 do
    k = d[0]
    if d[i] < 0 or d[0] > n-1 then
        return False
    end if
    for j = i + 1 to k − 1 do
        d[j] = d[j] − 1
    end for
end for
if d[n − 1] == 0 then
    return True
else
    return False
end if
```

时间复杂度分析：该算法首先需要对集合进行排序，其时间复杂度至少为 O($log(n)$)。在计算顶点的度数时，用了两层循环，n 个节点的度数最大为 $n-1$，则两层循环的时间复杂度为 O(n^2)。所以综合来看，该算法的时间复杂度为 O(n^2)。

2 Greedy Algorithm

There are n distinct jobs, labeled $J1, J2, \cdots, Jn$, which can be performed completely independently of one another. Each jop consists of two stages: first it needs to be preprocessed on the supercomputer, and then it needs to be finished on one of the PCs. Let's say that job J_i needs p_i seconds of time on the supercomputer, followed by f_i seconds of time on a PC. Since there are at least n PCs available on the premises, the finishing of the jobs can be performed on PCs at the same time. However, the supercomputer can only work on a single job a time without any interruption. For every job, as soon as the preprocessing is done on the supercomputer, it can be handed off to a PC for finishing.

Let's say that a schedule is an ordering of the jobs for the supercomputer, and the completion time of the schedule is the earlist time at which all jobs have finished processing on the PCs. Give a polynomial-time algorithm that finds a schedule with as small a completion time as possible.

问题分析与证明：先考虑最小情况。当只有一个作业时，一定要先在超算上先预处理，然后才能在 PC 上完成，其总时间 $T = p_i + f_i$。

当有两个作业时，假设 J_1 的 p_1 为 2，f_1 为 1，J_2 的 p_2 为 3，f_2 为 4。先执行 J_1，其运行总时间为 2+3+4=9。先执行 J_2，其运行总时间为 3+4=7。所以先执行 J_2。将 J_1 的 f_1 改为 5，则先执行 J_1，其总时间为 2+3+4=9，先执行 J_2，其总时间为 3+2+5=10。要先执行 J_1。

贪心策略为，每次选择在 PC 上执行最长的执行，每次将 f_i 中最大的放到超算上进行处理，然后将其分配到 PC 上继续计算。

现考虑一般情况，对上述策略进行正确性证明：首先我们将这 n 个作业按照 f_i 进行降序排序，不妨假设 J_i 的 PC 执行时间是最长的，则按照贪心策略执行 J_i。现假设最优结构中不优先执行 J_1，而执行某一个 $J_k(k > 1)$，而后再去执行 J_1。假设执行到 n 个作业的总时间为 T，则先执行 J_k 的总时间为 $\max(T + p_k + p_i + f_i, T + p_k + f_k)$，由于 f_i 一定大于 f_k，因此最长时间一定是 $T + p_k + p_i + f_i$。先执行 J_i，总时间为 $\max(T + p_i + p_k + f_k, T + p_i + f_i)$，由于 f_i 一定大于 f_k，所以 $T + p_k + p_i + f_i > T + p_i + p_k + f_k$ 且 $T + p_i + p_k + f_i > T + p_i + f_i$，即 $T + p_k + p_i + f_i > \max(T + p_i + p_k + f_k, T + p_i + f_i)$。所以无论怎样先执行 J_k 的时间总是比先执行 J_i 的时间长。即每次都需要去先执行在 PC 上时间最长的。

Greedy 2 Greedy 2

sort $f_1, f_2, ..., f_n$ as f[n] //降序排列 $f_1, f_2, ..., f_n$

return f[n]

时间复杂度分析：该算法需要对数组进行排序，其时间复杂度至少为 O($log(n)$)。