091M4041H - Assignment 3

胡鹏飞 计控学院 201828007329009

1 Greedy Algorithm

Given a list of n natural numbers d1, d2,...,dn, show how to decide in polynomial time whether there exists an undirected graph G = (V; E) whose node degrees are precisely the numbers d1; d2;...; dn. G should not contain multiple edges between the same pair of nodes, or “loop" edges with both endpoints equal to the same node.

1.1 Describe the basic idea in natural language AND greedy-choice property

首先，图 G 的边数为 e，那么图 G 所有顶点的度数之和为 2e，即必然为偶数，这是构成简单图的充要条件，所以先看 d1, d2, ……dn 之和是否为偶数。若为偶数，则继续判断，使用贪心算法的话，很容易可以想到将 d1, d2……dn 按照从大到小的顺序进行排列，若排序之后第一个为 dk，若 dk 大于你，则必然存在环或者重复边，若 dk 小于 n 则继续判断，此时可以认为这个顶点与其后 dk 个顶点有连接，所以其后 dk 个顶点每个都需要减 1，减完之后重新排序，重复以上操作，若中间某个顶点度数小于 0，则必然存在孤立顶点，无法构成无向图，直至最后一个顶点，若为 0，则可以构成无向图。

1.2 Pseudo-code

```
ExistGraph(d[1……n])
    if((d1+d2+…+dn)%2=1) then
        return False
    end if
    while(n)
        sort(d)
        d1 = d[0]
        if d1 > n then
            return False
        end if
        for i = 1 to d1 then
            d[i] -= 1
            if d[i] < 0 then
                return False
            end if
        end for
        d[0] = 0
        n -= 1
    end while
    if(d[0]!=0) then
        return False
    else
        return True
```

1.3 Prove the correctness

首先，算法排除了三种情况，这三种情况都是不能构成简单图的情况，上述已经说明。其次，算法在循环中每次排序选取最大度数的点，这样就可以使后面有足够多的点可以抵消该点的度数，使该顶点连接其他 d1 个节点，之后就不再考虑该顶点，然后在剩下的顶点中找度数最大的顶点，给它分配第二大的度数，即与 d2 个顶点连接，如此类推，直到分配完所有度数，综上所述，算法无误。

1.4 Analyse the complexity

首先，需要循环的次数最多为 n 次，复杂度为 O(n)，每次循环里需要排序，复杂度为 O(nlogn),所以总的时间复杂度为：

$$T(\mathrm{n}) = O(\mathrm{n}^2\mathrm{logn})$$

3 Greedy Algorithm

Given two strings s and t, check if s is subsequence of t?
A subsequence of a string is a new string which is formed from the original string by deleting some (can be none) of the characters without disturbing the relative positions of the remaining characters. (ie, "ace" is a subsequence of "abcde" while "aec" is not).

3.1 Optimal substructure and DP equation

根据贪心，目标串前面的字符匹配的位置越靠前，源串后面的字符越多，匹配上的概率越大，所以定义 s 和 t 两个字符串的索引，如果发现 t 的索引值和 s 的索引值相等，移动子串 s 的索引，继续比对下一个目标索引值，如果最后 s 的索引和 s 的长度相等，那么 s 即为 t 的子串。

3.2 Pseudo-code

```
IsSubsequence(s,t)
    sIndex = 0;
    tIndex = 0;
    while (sIndex < s.length() && tIndex < t.length())
        if t[tIndex] == s[sIndex] then do
            sIndex++
        end if
        tIndex++
    end while
    if sIndex != s.length() then
        return False
    else
        return True
```

3.3 Prove the correctness

两个字符串的索引意味着将问题分解成了子问题，子问题中是新的源串和目标串，在源

串中找到了目标串的第一个字符，那么在源串中继续找到目标串下一个字符的概率就会增大，就成了下一个子问题，后面的子问题中不需要考虑之前的子问题的内容，即后面的子问题不需看到之前的问题，算法无误。

3.4 Analyse the complexity

算法要遍历一次源串 t，即

$$T(\mathrm{n}) = O(\mathrm{n})$$

4 Greedy Algorithm

Given a rope whose length is n, please cut the rope to m parts to get the maximum product of the length of each part ∏l1+l2+::+lm=n l1 * l2 * …* lm.For example, if a rope with length 8, when we cut it to 2, 3, 3, we can get the maximum product 18.

4.1 Optimal substructure and DP equation

在剪绳子中，如果绳子的长度大于 5，则每次剪出的长度为 3 的绳子。如果剩下的长度仍然大于 5，则接着剪出一段长度为 3 的绳子，重复这个步骤，即尽可能多的剪出长度为 3 的绳子，直到剩下的长度小于 5，当剩下的长度为 4 时，把绳子剪成 2 的绳子。

4.2 Pseudo-code

```
FindMaxPreouct(length)
    if length < 2 then
        return 0
    end if
    if length == 2 then
        return 1
    end if
    if length == 3 then
        return 2
    end if
    m = length / 3
    if length - m * 3 == 1
        m -= 1
    end if
    n = (length - m * 3) / 2;
    return pow(3, m)*pow(2, n);
```

4.3 Prove the correctness

首先，当 n=1, 2, 3 时，问题很容易求解；当 n=4 时，剪出一个 3 和一个 1 不如两个 2 乘积大；当 n>=5, 3(n-3)>=2(n-2)>n，即分出 3 比分出 2 乘积要大，更大于不分，所以应该不断分出 3，这样乘积最大，length 整除 3 后余数如果是 1，说明多剪了一个 3，应该拿出来

一个 3（即 m-1）和剩下的 1 组成 4,算法无误。

4.4 Analyse the complexity

因为只对 length 进行简单运算，所以时间复杂度为：

$$T(\mathrm{n}) = O(1)$$