算法设计与分析 Week 9

第九周作业-Solution

Lecturer: 杨启哲 Last modified: 2024 年 11 月 13 日

1. 如果图含有负权重，Prim 算法是否能给出正确的最小生成树？请证明你的结论。

   解答. Prim 算法可以给出正确的最小生成树，不受负权重的影响。事实上，假设 n 个顶点的图 G 存在负权重，令 c 是权重最小的边的权重，那么我们可以构造一个新的图，顶点和边与原图相同，但是每条边的权重都加上 $-c+1$，那么这个新图 G' 的所有权重都是正的，并且 G' 的最小生成树的权重正好比 G 的最小生成树的权重多 $(n-1)(1-c)$，即 G' 的最小生成树恰好也是 G 的最小生成树。因此，我们可以在 G' 上运行 Prim 算法，得到的最小生成树正好是 G 的最小生成树，即负权重不影响 Prim 算法的正确性。 □

   > Remark 0.1
   >
   > 大家可以与 Dijkstra 算法作为对比，为什么几乎是同样的处理方式，但是 Dijkstra 算法不能处理负权重的情况。其主要原因在于任何一颗生成树它都必须是 $|V|-1$ 条边。当边数是固定的时候，我们就可以通过加一个常数来保证所有的边都是正的，从而形成一个一一映射保证最后的答案。

2. (最大生成树) 请给出一个算法，求出一个无向图的最大生成树，即权重最大的生成树。

   解答. 利用上一问的结论我们可以很快获得一个处理方式：

   (1) 将图中所有边的权重取相反数。
   (2) 运行 Prim 算法，得到该图的最小生成树，即为原图的最大生成树。

   □

3. (长途旅行) 假设您准备开始一次长途旅行。以 0 英里为起点，一路上一共有 N 座旅店，距离起点的英里数分别为 $a_1 < a_2 < \ldots < a_n$。旅途中，您只能在这些旅店中停留，当然在哪停留完全由您决定。最后一座旅店 a_N 是您的终点。

   理想情况下，您每天可以行进 200 英里，不过考虑到旅店的实际距离，有时候可能还达不到这么远。假设某天您走了 x 英里，那么您将受到 $(200-x)^2$ 的惩罚。您需要计划好行程，使得总的惩罚-每天所受惩罚的总和最小。请给出一个高效的算法，来确定一路上最优的停留位置。

   解答. 记 $L[k]$ 表示到达第 k 座旅店所受到的最小惩罚，初始我们令 $L[0]=0$，即为起点。则我们可以得到如下的关系式：

   $$L[k] = \min_{j<k, a_k - a_j \leqslant 200} L[j] + (200 - a_k + a_j)^2$$

根据上述公式更新 $L[k]$ 即可获取最终答案。注意到 200 是一个常数，从而每次循环执行的比较次数也是常数 (只跟 200 有关)，所以算法的时间复杂性为 $O(n)$. □

4. (最长回文子序列) 如果一个子序列从左向右和从右向左读都一样，则称之为回文。例如，序列：

$$A, C, G, T, G, T, C, A, A, A, A, T, C, G$$

有很多回文子序列，如 A, C, G, C, A 和 A, A, A, A。请设计一个 $O(n^2)$ 时间的算法，对于输入一个长度为 n 的序列，找出其最长回文子序列的长度。

解答. 我们可以利用动态规划的思想来解决这个问题。x 需要注意的是，因为是回文子序列，所以并不一定连续。因此记 $L[i][j]$ 为序列 $A[i], \ldots, A[j]$ 的最长回文子序列的长度，则我们有如下递推关系：

$$L[i][j] = \begin{cases} 1 & i = j \\ 2 & i = j-1, A[i] = A[j] \\ L[i+1][j-1] + 2 & i < j-1, A[i] = A[j] \\ \max\{L[i+1][j], L[i][j-1]\} & i < j, A[i] \neq A[j] \end{cases}$$

根据上述递推关系可以构造如下算法：

(1) 初始化 $L[i][i] = 1$ 和所有的 $L[i][i+1]$。

(2) 根据 $L[i][j]$ 中 $j - i$ 的大小从 2 到 $n-1$ 开始循环，依次根据上述关系更新每组的 $L[i][j]$。

简单的伪代码如下：

最长回文子序列

输入: 序列 $A[1, \ldots, n]$
输出: 最长回文子序列的长度

```
for i = 1 to n do
    L[i][i] ← 1
end for
for i = 1 to n − 1 do
    if A[i] = A[i + 1] then
        L[i][i + 1] ← 2
    else
        L[i][i + 1] ← 1
    end if
end for
for k = 2 to n − 1 do
    for i = 1 to n − k do
        j ← i + k
        if A[i] = A[j] then
```

```
            L[i][j] ← L[i + 1][j − 1] + 2
        else
            L[i][j] ← max{L[i + 1][j], L[i][j − 1]}
        end if
    end for
end for
return L[1][n]
```

算法需要进行 $n \times n$ 的一个循环，每次循环的操作次数是常数次，所以时间复杂度为 $O(n^2)$。 □

5. 我们考虑一个新的方法来得到最小生成树：

(1) 请证明如下性质：选择图中任何一个环，删除其中权重最大的边，得到的图的最小生成树仍然是原图的最小生成树。

(2) 考虑如下的算法：

算法:MST-DeleteCycle

(i) 将图中的边按照权重从大到小排序。

(ii) 按排好的顺序考虑每一条边，如果这条边在图中形成了环，则删除这条边。

(iii) 直到图中没有环为止。返回剩下的图。

请证明算法的正确性。

(3) 注意到，判断一条边是否在某个圈里我们已经在上次作业里出现过，该目标可以在线性时间内完成。基于此信息请分析算法的时间复杂度。

解答.

(1) 反设结论不成立。即存在一个圈 φ 删去其权重最大的边 e 以后，得到的最小生成树不相同。令删掉 e 后的图为 G'，相应的最小生成树为 T, T'。注意到 G 和 G' 具有相同的顶点，从而 T' 也是 G 的生成树，从而 $e \in T$, 否则 T 不是最小生成树。

现在考虑 $T - \{e\}$，由最小生成树的性质，其必然是不连通的，并且其将 G 划分成了 V_1, V_2 两个点集，满足 e 是其中一条横跨的边。由于 φ 是一个包含 e 的圈，因此 φ 中必然还存在一条横跨 V_1, V_2 的边 e'，由假设 e' 的权重小于 e，从而 $T - \{e\} \cup \{e'\}$ 是一个更小的生成树这与 T 是最小生成树矛盾，因此结论成立。

(2) 令每一轮后剩余的图为 G_i，由上一问的结论我们有：

Lemma 0.2

G_i 的最小生成树是 G_{i-1} 的最小生成树。

注意到算法最终返回的也是 G 的生成树，因此算法的正确性得证。

(3) 令 m 表示边的个数，n 表示顶点的个数。整个算法的时间如下:

(i) 排序的时间复杂度为 $O(m \log m)$。

(ii) 每次判断是否形成圈的时间复杂度为 $O(m)$。循环至多执行了 $m - n + 1$ 次。

因此整个算法的时间复杂度为 $O(m^2)$。

□