清华大学软件学院

算法分析与设计基础

2020 年春季学期

作业 **9**

吴佳龙　班级：软件 83　学号：2018013418　May 21, 2020

9.1. (Exercises 34.3-2)

证明. $L_1 \le_p L_2$，则存在可多项式时间计算的 f，$x \in L_1 \Longleftrightarrow f(x) \in L_2$。

$L_2 \le_p L_3$，则存在可多项式时间计算的 g，$x \in L_2 \Longleftrightarrow g(x) \in L_3$。

则存在 $g \circ f$ 可多项式时间计算，且 $x \in L_1 \Longleftrightarrow f(x) \in L_2 \Longleftrightarrow g(f(x)) \in L_3$，因此 $L_1 \le_p L_3$。

$\le_p$ 具有传递性。 □

9.2. (Exercises 34.5-7)

证明. 最长简单回路问题是一个优化问题，转化为对应的判定问题为输入 x 是否属于

$$L = \{(G, k) \mid G \text{ 中存在长度为} k \text{ 的简单回路}\}$$

首先该问题是 NP 的，因为给定一个证书 y 为 G 中的一列长为 n 的顶点序列可以在 $O(n)$ 的时间内判定是否 $n = k$，该道路是不是简单的，以及是不是回路：只需要逐一检查每两个相邻的点是否有边相连，起终点是否相等，每个点是否仅出现一次。

其次，该问题是 NP-Hard 的。因为对于已知的 NPC 问题：哈密顿回路问题，存在 f 定义为 $f(G) = (G, n)$，其中 n 为 G 中点的个数。由哈密顿回路的定义，输入 G 满足哈密顿回路问题，当且仅当 $(G, n) \in L$。因此哈密顿回路问题可规约为该问题。

综合以上，该问题是 NPC 的。 □

9.3. (Problems 34-1)

(a) 独立集问题相关的判定问题的形式描述为判定输入 x 是否属于

$$L = \{(G, k) \mid G \text{ 中存在大小为} k \text{ 的独立集}\}$$

现证明它是 NPC 的。

证明. 首先它是 NP 的，因为给定一个证书 $y = V' \subset V, |V'| = k$，可以在 $O(km)$ 的时间判定他是否是独立集，只需要对于每一条边，判断它的两个顶点是否都在 V' 中即可。其次它是 NP-Hard 的，因为他可以从团问题规约。这是因为 G 存在一个大小为 k 的团，等价于它的补图 $\bar{G}$ 存在一个大小为 k 的独立集。

若 V' 是 G 中大小为 k 的团，则任意 $u, v \in V', (u, v) \in E$，则在 $\bar{G} = (V, E^c)$ 中 $(u, v) \notin E^c$，V' 是 $\bar{G}$ 中的独立集。

若 V' 是 $\bar{G}$ 中大小为 k 的独立集，则任意 $u, v \in V', (u, v) \notin E^c$，则在 $G = (V, E)$ 中 $(u, v) \in E$，V' 是 G 中的团。

综合以上两点，该问题是 NPC 的。 □

(b) 算法伪代码如下：

```
1  FIND−INDEPENDENT−SET(G):
2      n = |V|
3      k = 0
4      for i=1 to n
5          if HAS−INDEPENDENT−SET(G,i):
6              k = max(k,i)
7      G’ = G
8      for (u,v) not in E:
9          add (u,v) to G’
10         if not HAS−INDEPENDENT−SET(G’,k):
11             delete (u,v) from G’
12     independent−set = {u | deg(u)<n−1 in G’}
13     return independent−set
```

其中 3-6 行用于求得最大独立集的大小 k，8-12 行用于求出最大独立集。该算法是正确的，因为如果向图中加入一条边 (u, v) 导致最大独立集的大小变小了，这说明对于所有可能的最大独立集 u, v 在其中，否则就说明 u, v 都在最大独立集中不是必须的。最终得到的图 G' 中所有没有边相连的两点都在最大独立集中，因此最大独立集是所有度数不满的点构成的集合。

若将 HAS-INDEPENDENT-SET 的复杂度看做 $O(1)$，则该算法的复杂度是 $O(|V| + |E|)$。

(c)

Lemma 1. 每个点度数为 *2* 的连通图必然构成一个环。

证明. 设图中点的个数为 n，每个点度数为 2，则边的个数也为 n，则存在一条边 e，剩余的边构成一棵树。该树必然只有两个叶子结点，否则即使加入一条边，也不能使所有点的度数为 2。则该树退化成一条链，且链的两端就是 e 的两端，加入边 e，成为一个环。 □

Corollary 1. 每个点度数为 *2* 的图由若干个不连通的环构成。

Lemma 2. 大小为 n 的环的最大独立集是 $\lfloor n/2 \rfloor$。

证明. 在环上，在独立集中的两个点之间至少间隔了一个不在独立集中的点，因此独立集大小至多为 $\lfloor n/2 \rfloor$。且该上界可以取到，只要从任意一个点出发，每隔一个点选择进入独立集即可。 □

由以上引理和推论，求解每个点度数为 2 的图的最大独立集的**算法描述** 为：枚举每个没有被遍历过的点，从该点出发遍历整个环，设环的长度是 c_i，则最大独立集的大小增加 $\lfloor c_i/2 \rfloor$，在环上每隔一个点选择进入独立集。

该算法的复杂度是图遍历的复杂度，为 $O(|V| + |E|) = O(|V|)$，其中 $|E| = 2|V|$。

(d)

Lemma 3. 对于二分图，最小顶点覆盖大小等于最大匹配数。

证明. 首先求得二分图 $((X, Y), E)$ 的一个最大匹配 M，其中 X 表示左部点集，Y 表示右部点集。

从右部的每一个未匹配点出发，交替地通过未匹配边、匹配边、未匹配边、$\cdots$、匹配边的顺序遍历整个图。最后一定会在匹配边终止，因为如果以未匹配边结尾，将整条链上匹配边、未匹配边调换成未匹配和匹配，则匹配数会增大，与最大匹配矛盾。注意遍历的过程中如果有多条非匹配边可走，则都要走一遍。

记这样的遍历过程经过的所有点构成点集 S，令 $T=(S\bigcap X)\bigcup(S^c\bigcap Y)$。$T$ 具有如下性质：

i. $|T|=|M|$，即最大匹配数。这是因为：由于从右部走向左部的时候经过的都是非匹配边，$S\bigcap X$ 都是匹配点；又由于右边的未匹配点一定在 S 中，$S^c\bigcap Y$ 也都是匹配点；与 $S\bigcap X$ 匹配的点并上 $S^c\bigcap Y$ 就是右部所有的匹配点，因此两个集合的大小之和为匹配数。

ii. T 是一个顶点覆盖。所有的边 (u,v) 根据两个端点是否被遍历到，可分为三种：
   - $u\in S, v\in S$，这种边被 $S\bigcap X$ 覆盖；
   - $v\notin S$，这种边被 $S^c\bigcap Y$ 覆盖；
   - $u\notin S, v\in S$，这种边不存在，这是因为：若该边是匹配边，由 $v\in S$，这条边一定被遍历过，$u\in S$ 且先于 v 被遍历，矛盾；若该边是非匹配边，根据匹配规则，$v\in S$ 一定会沿着这条边去遍历 u。

iii. T 是最小的顶点覆盖集。若 $|T|<|M|$，T 中都是匹配点，无法覆盖匹配中的 $|M|$ 条边。

综合以上三条性质，结论得证。 □

Lemma 4. 对于二分图，最大独立集大小等于顶点数减去最小顶点覆盖大小。

证明. 取二分图的一个最小顶点覆盖 T，则 $V\backslash T$ 是一个最大独立集。

首先证明他们独立，若 $u\notin T, v\notin T, (u,v)\in E$ 则与顶点覆盖矛盾。

再证最大性，我们已知了顶点覆盖与独立集一一对应，由于 T 是最小顶点覆盖，则 $V\backslash T$ 自然是最大的。 □

由以上引理，求解二分图的最大独立集的**算法描述** 为：首先求得最大匹配，复杂度为 $O(VE)$；再根据 Lemma 3 中定义的遍历过程，构造 T，复杂度为 $O(V+E)$；在由 Lemma 4，将 T 取补集得到最大独立集。

该算法的总复杂度为 $O(VE)$。