TSINGHUA UNIVERSITY

Homework #3

Student name:

Course: 算法分析与设计 – Professor: 王振波

Due date: *2020 年 10 月 8 日*

3.2 Give an algorithm to detect whether a given undirected graph contains a cycle. If the graph contains a cycle, then your algorithm should output one. (It should not output all cycles in the graph, just one of them.) The running time of your algorithm should be $O(m+n)$ for a graph with n nodes and m edges.

我们不妨假设这是一个连通图 $G=(V,E)$(否则下列算法可将其自动分为几个连通图)。我们给出如下算法：

```
任意选取一个点 x ∈ G，将其记为 L_0，初始化 k = 1, V̄ = V − x；
while V̄ ≠ Φ do
    在 L_{k−1} 中依次选取节点 y，在 V̄ 中扫描所有与其相连的节点 ychild_i，记录下对应的边，并将 ychild_i 加入到 L_k 中，令 V̄ = V̄ − ychild_i
    取出 L_{k−1}, L_k 中所有的边 e_{k−1,k}，与 E 中关于此两层节点的所有边 E_{k−1,k} 做对比
    if e_{k−1,k} = E_{k−1,k} then
        k = k + 1
    else
        记录下一个边 e ∈ E_{k−1,k} − e_{k−1,k}，return find cycle
    end if
end while
return No cycle
```

易知此算法和 BFS 算法复杂性相同，均为 $O(m+n)$。

下面说明算法的正确性：

- 若返回了 No cycle，则说明 E 中所有的边都在 BFS 算法生成的树中，这当然没有环，
- 若返回了 find cycle，则说明 E 中有不在树里的边，我们知道这种边的节点 (y,z) 在树中的层数不超过 1，无论如何都会导致环的产生。那么从根 x 出发到 y，再从 (y,z)，

再从 (z, x) 构成一个环，我们只需要在这个路径中去掉重复经过的点，就构成了一个真正的环。

■

> **3.8** Claim: There exists a positive natural number c so that for all connected graphs G, it is the case that
> $$\frac{\mathrm{diam}(G)}{\mathrm{apd}(G)} \leq c$$
> Decide whether you think the claim is true or false, and give a proof of either the claim or its negation.

假设不正确！

考虑这样一个图：共 n^2+n 个点，其中 n 个点 $a_1, a_2, \cdots, a_n$ 顺次相连，而另外 n 个点加上 a_1 构成一个完全图。

可知 $\mathrm{diam}(G) = n$，而

$$\mathrm{apd}(G) = \left(\frac{n^3-n}{6} + \frac{\left(n^2-1\right)n^2}{2} + \frac{n^3(n+1)}{2}\right) / \left(\frac{\left(n^2+n\right)\left(n^2+n-1\right)}{2}\right)$$

取极限 $n \to \infty$ 可知 $\lim_{n\to\infty} \mathrm{apd}(G) = 2$，于是 $\frac{\mathrm{diam}(G)}{\mathrm{apd}(G)}$ 的阶数可达 $\Theta(n)$，不可能小于一个正整数。 ■

> **4.4** Give an algorithm that takes two sequences of events- S' of length m and S of length n, each possibly containing an event more than once-and decides in time $O(m+n)$ whether S' is a subsequence of S.

我们给出如下算法：

```
1: for i = 1 : m do
2:     从 S' 中取出第 i 个元素 a_i，并在 S 中找到第一个等于 a_i 的元素，若没找到则 return False
3:     从 S 中删除 a_i 之前的所有元素
4: end for
5: return True
```

我们一共比较了 n 次，加上我们从 S' 中取了 m 次，总共时间复杂度为 $O(m+n)$

如果算法返回 True，那么我们每次在 S 中找的 a_i 构成了 S'，这说明 S' 是 s 的子列；如果算法返回 False，反证法易知 S' 不是 S 的子列。 ■

4.6 What's the best order for sending people out, if one wants the whole competition to be over as early as possible? More precisely, give an efficient algorithm that produces a schedule whose completion time is as small as possible.

假设 $t_i = a_i + b_i$，其中 a_i 代表第 i 个人游泳的时间，b_i 代表第 i 个人其余两项的总时间，我们按照 b_i 从大到小排序的顺序安排同学出发。

不妨假设 $b_1 \geq b_2 \geq \cdots \geq b_n$，我们给出两种证明方式。

证明 1: 我们假设在我们的算法中最后到达的是第 k 号同学，则总用时 $t = \sum_{i=1}^{k} a_k + b_k$，而另一排序中在前 k 个人中最后出发的是 m 号同学 (则 $m \leq k$)，可见 m 号同学到达终点的时间至少为 $t' = \sum_{i=1}^{k} a_k + b_m$，由假设可知 $t \leq t'$。从而我们的算法给出了最短时间。

证明 2: 如果存在 $i < j$，使得第 i 位同学在第 j 位同学之后出发，即有一对逆序数 (j, i)。我们考虑交换此二人的顺序 (i, j)，容易知道第 i 位同学到达的时间提前了；而在顺序 (j, i) 中，i 号同学到达终点的时间为 $t = A + b_i$，其中 A 代表在 i 之前 (包括 i) 的游泳时间之和，在顺序 (i, j) 中，第 j 位同学到达的时间为 $t' = A + b_j$，可见 $t' \leq t$。

于是交换逆序数后，总时间不会增加，所以我们在经过交换逆序数后可将序列变为算法中的序列，且总时间更小了，所以我们的算法给出了最短时间。 ∎