

Homework #3

Student name:

Course: 算法分析与设计 – Professor: 王振波

Due date: 2020 年 10 月 6 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:

```
1: 任意选取一个点  $x \in G$ , 将其记为  $L_0$ , 初始化  $k = 1, \bar{V} = V - x$ ;  
2: while  $\bar{V} \neq \Phi$  do  
3:   在  $L_{k-1}$  中依次选取节点  $y$ , 在  $\bar{V}$  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  $\bar{V} = \bar{V} - ychild_i$   
4:   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  
5:   if  $e_{k-1,k} = E_{k-1,k}$  then  
6:      $k = k + 1$   
7:   else  
8:     记录下一个边  $e \in E_{k-1,k} - e_{k-1,k}$ , return find cycle  
9:   end if  
10: end while  
11: 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

$$\frac{\text{diam}(G)}{\text{apd}(G)} \leq c$$

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, \dots, 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 $\text{diam}(G) = n$, 而

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

取极限 $n \rightarrow \infty$ 可知 $\lim_{n \rightarrow \infty} \text{apd}(G) = 2$, 于是 $\frac{\text{diam}(G)}{\text{apd}(G)}$ 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.

我们按照每个人所花的总时间 t_i , 从长至短依次出发。

证明: 不妨假设 $t_1 \geq t_2 \geq \cdots \geq t_n$, 且 $t_i = a_i + b_i$, 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。我们假设在我们的算法中最后到达的是第 k 号同学, 而另一排序中在前 k 个人中最后出发的是 m 号同学 ($m \leq k$), 可见 m 号同学到达终点的时间一定不小于我们算法中的第 k 号同学。从而我们的算法给出了最短时间。 ■