算法设计与分析 (2018 年春季学期)

第四次作业参考答案

1 对下面的每个描述，请判断其是正确或错误，或无法判断正误。对于你判为错误的描述，请说明它为什么是错的。(每小题 5 分，共 20 分)

1. $P \subset NP$；
2. $NPC \cap P = \emptyset$；
3. 若 SAT 问题可以用复杂度为 $O(n^9)$ 的算法来解决，则所有的 NP 完全问题都可以在多项式时间内被解决；
4. $UNSAT \in NP$（$UNSAT$ 问题是指：给定一个布尔表达式 ϕ，判断是否对其中变量的所有取值，ϕ 的值均为 $false$）

解：

1. 正确；
2. 无法判断；
3. 正确；
4. 无法判断。

2 最小生成树问题 (25 分)

给定一个无向连通图 $G = (V, E)$，其中每条边的权值只可为 1 或 2。请设计一个时间复杂度为 $O(|V| + |E|)$ 的算法来求 G 的一棵最小生成树，并验证其时间复杂度。(注：你可以提出一个新算法或修改课堂上讲过的算法。)

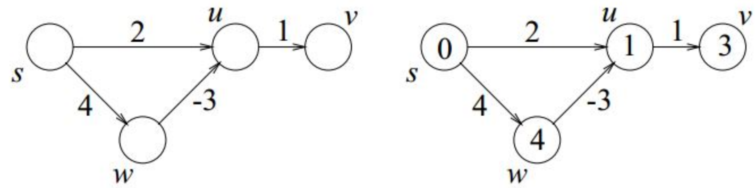
解：
该问题有多种解法，这里仅给出一种解法的主要思想。该解法的思路是使用一个更简单的数据结构来代替 $Prim$ 算法中用到的优先队列 (*priority queue*) 从而使得队列的插入和查找操作的复杂度都是 $O(1)$，而不是 $O(\log n)$。具体做法是使用两个链表来代替优先队列。在 $Prim$ 算法的运行过程中，用链表 L_1 来存储所有长度为 1 的边，L_2 来存储所有长度为 2 的边，这样可以很容易的实现 *Extract-Min()* 和 *Decrease-Key()* 操作。这两个函数的具体实现留作小练习，这里不再给出。

3 最短路问题 (25 分)

请给出一个边权可以为负的有向图实例，使得 $Dijkstra$ 算法在该图上无法得到正确的结果。并解释允许边权为负的情况下 $Dijkstra$ 算法不再正确的原因。(注：给出的实例应保证该有向图不存在权值和为负的环。)

解：
这个问题的关键就是在 $Dijkstra$ 算法的执行过程中，要在某个点的真实的最短路被求出之前将它加入已选集合。考虑如下图所示的实例，在计算到 s 的单源最短路的过程中，第一步我们

会得到 $d[u]=2,d[w]=4$，因此我们先选择将 u 加入已选集合，从而得到 $d[v]=3$，下一步就会把 v 加入已选集合，最后再把 w 加入集合。这样我们最终得到 s 到 v 的最短路长度为 3，而事实上，路径 $<s,w,u,v>$ 的长度更短，为 2。



在允许边权为负的情况下 $Dijkstra$ 算法不再正确的原因如下：回忆在算法正确性证明过程中的第二种情况，令 y 是 s 到 u 的最短路上的任意一点且 $y \neq u$，在课上给出的证明过程中，我们断言因为 y 是 s 到 u 的最短路上的任意一点，所以一定有 $\delta(s,y)<\delta(s,u)$。这在所有边权为非负的时候确实是正确的，但如果允许边权为负的话，该性质将不再成立。因此，此时 $Dijkstra$ 算法的正确性无法保证。

4　环路问题 (30 分)

给出一个联通无向图 $G=(V,E)$，请设计一种尽可能高效的算法来判断 G 中是否有环。如果有，请输出任意一个环 (按顺序给出环上的每个顶点)。请解释算法的正确性并分析算法的时间复杂度。

解：

选出任意一点 $s \in V$，从该点开始在图 G 上进行深度优先搜索 (DFS)。当搜索过程中遇到一条反向边 (u,v)，我们就可以得出结论：G 中存在环。接下来从 u 开始，输出当前搜索栈中记录的所有点，直到 v 结束，这就是 G 中的一个环。

正确性：无向图中的边要么是树边，要么是反向边 (定理证明见 Lecture09)，如果图中有环，那么一定存在一条反向边。如果图中没有环，意味着该图是一个树结构，不存在反向边。

时间复杂度：该算法的时间复杂度为 $O(|V|)$。因为在该算法的执行过程中所有的点至多被访问一次，因此我们最多也仅会访问 $O(|V|)$ 条边。(如果此处复杂度分析为 $O(|V|+|E|)$ 会扣 2 分)

算法的伪代码可参考 **Algorithm 2**：

Algorithm 1 $Visit(u)$

$color[u] \leftarrow$ GRAY;
for $v \in Adj(u)$ **do**
　if $color[v]=$ WHITE **then**
　　$pred[v] \leftarrow u$;
　　$Visit(v)$;
　else
　　if $v \neq pred[u]$ **then**
　　　$begin \leftarrow u$;
　　　$end \leftarrow v$;
　　　return ;
　　end if
　end if
　if $begin \neq$ NULL **then**
　　return ;
　end if
end for

Algorithm 2 $Cycle(G)$

```
for u ∈ V do
    color[u] ← WHITE;
    pred[u] ← NULL;
end for
begin ← NULL;
end ← NULL;
Visit(1);
if end = NULL then
    output No Cycle;
else
    while begin ≠ end do
        output begin;
        begin ← pred[begin];
    end while
    output end;
end if
```