21.2-3 假设每次结点更新和查询的实际代价为 1，而支付的代价为 2。
对于 MAKE-SET 操作，创造一个新对象支付代价为 1，剩余信用总是不小于 0，所以 n 个 MAKE-SET 操作总的摊还代价为 O(2n)，即平均摊还代价为 O(1)。
对于 FIND-SET 操作，查询一个元素代价为 1，剩余信用总是不小于 0，所以 n 个 FIND-SET 操作总的摊还代价为 O(2n)，即平均摊还代价为 O(1)。
对于 UNION 操作，对每个结点最多 lgn 次更新，更新一次代价为 1，UNION 查询需要 2 次，查询实际代价为 2，UNIOIN 操作最多 n-1 次，注意到剩余信用总是不小于 0，总摊还代价为 O(2nlgn)+O(2(n-1))=O(2nlgn+2(n-1))，平均摊还代价为 O(2nlgn+2(n-1))/(n-1)=O(lgn)。

21.3-2

```
FIND-SET(x) :
    Tp = x
    While Tp ! = Tp.p :
        Tp=Tp.p
    P=x
    While P != P.p
        Temp=P.p
        P.p=Tp
        P=temp
    Return Tp
```

21.2-5
a. 直接证明 u.d 是源 s 到 u 的最近距离即可，设路径 L1 满足 L1(s,u)=d1，更短距离 L2 满足 L2(s,u)=d2<d1，此时距离更短的点必然会先进队列，同时也先出队列，即 u.d1 会把 u.d 先赋值，等到 d2 希望赋值 u.d 的时候 u 已经染成黑色。所以由上必然有最短距离先将 u.d 赋值。即 u.d 即为 dist(s,u)。而邻接表的一个链表顺序变化不影响图结构，所以 u.d 不受影响。
b. 在 22.3 图中，当程序迭代到将 t 与 x 纳入队列时，如果两者在 w 对应的链表中位置不同，结点 u 是广度树中的父节点会发生变化，如果 t 在 x 前则 u 的父节点会先指向 t，否则 u 的父节点会指向 x。

22.3-8 考虑两颗以 A,B 为根有向树（不妨两者深度都为 3），树中的边只能从父节点指向孩子结点，两个树根之间存在有向边 AB，则两颗有向树和有向边 AB 构成有向图 G，考虑以 B 为根的树深度为 3 的一个结点 C，易知从 A 到 C 一定存在一条有向路径，且我们发现从 A 开始深度搜索时，A.d<C.d，所以题中条件满足，然而易知图 G 此时生成的深度搜索森林恰好为以 A,B 为根有向树（不包括边 AB），所以 C 并不是 A 的后代。

23.1-3 反证，假设(u,v)边不是横跨图 G 的某个切割的一条轻量级边，则把生成树去掉边(u,v)，换成关于(u,v)的切割的轻量级边，可知生成树依然是生成树，然而树的总权重减少，这与原来生成树是最小生成树矛盾。所以原命题成立。

23.2-8 成立，对算法中划分的两个点集合 A，B，任意一个图 G 的生成树由这两个点集合 A、B 的生成树、一条横跨 AB 的边组成，为了使得图 G 的生成树总权重最小，即要求 A 中的生成树权重最小、B 中生成树权重最小、且横跨边权重也是最小，所以算法由递归得到的解是最优的解。