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。