ADA hw3 hand written　　B05902022 張雅信

problem 5

(1)
the amazing hash table stores the node's id, its left and right children
use postorder to traverse the tree
(for each node, first traverse its left children, then right children, then itself)

```
for each node v,
        if (v is in the hash table)
                link v's parent to the pointer in the hash table
                delete v
        else
                insert v into the hash table
```

(2)
In the algo, a node will be insert into the hash table only if there is no node equal to it in the hash table.
That is, all node in the hash table are different(not equal). ..(i)
All other nodes are merged. (ii)
By (i) and (ii), all nodes that are not merged are not equal,
Thus the number of nodes is minimized.

(3)

```
for each node V
        if (v is in the hash table)                               =>search:O(1)
                link v's parent to the pointer in the hash table  => O(1)
                delete v                                          => O(1)
        else
                insert v into the hash table                      =>O(1)
```

thus time complexity = V*O(1) =O(V)

problem 6

(1)
to connect two disjoint set, we only need to add one edge.
to connect all V vertice, we only need to add V-1 edges,
since adding each edge will reduce the number of disjoint sets by one.

if T is cyclic, then we must be able to cut one of the edges in the cycle,
without dividing the graph into two disjoint subgraph.

(2)
***
the statement is correct only when every vertex in the region can be reached from any other vertex in the region without leaving the region in the MST of the whole graph.
***
prove by contradiction:
Let S=(V1,E1) be a subtree of the MST of the whole graph.
if S is not the optimal subtree, then
there exists S'=(V1,E2) that is the MST of the subgraph,
for every edge e that is in T but not in S, add e to S', then the new tree T' will have less weight then T, which forms a contradiction (T is not MST).

(3)
prove by contradiction:
if the result (T1) of Alice's algo is not MST (T2), then there exist an edge e1 that can be replaced by another edge e2 with weight(e1)>weight(e2)
that is,
when encountering e1, T-{e1} is not connected ..(i)

To maintain the result to be a tree, e1 and e2 should connect the same two subtree, and T1+{e2}=T2+e{1} is connected. Also, T1 and T2 are connected.
This forms a contradiction since
when (i) happens, T ⊂ T2+{e1} => T-{e1} ⊂ T2 is connected.

(4)
in a complete graph, O(E) = O(V^2)
Prim's algo:
T(V) = O(E+VlogV) = O(V^2)

Kruskal's Algo:
sort the edges: O(ElogE)
picking edges: O(E)
T(V) = O(V^2logV)

(5)
in a planar graph, O(E) = O(V)
Prim's algo:
T(V) = O(E+VlogV) = O(VlogV)

Kruskal's Algo:
sort the edges: O(ElogE)
picking edges: O(E)
T(V) = O(VlogV)