Homework 7

Problem 1: How many bits do you need to store the size and height of the trees in the Union Find data structure? Give your answer for different implementations (e.g., arbitrary unions, unions by size, union by height, path compression etc.).

The size for every implementation is upper bounded by N, the total number of elements in the set. This corresponds to the tree which includes all the elements. This requires $O(\log N)$ bits for storing the size.

In case of arbitrary union, $N-1$ unions may cause a tree as a chain of N nodes. So, the height of a tree is N in worst case. Thus, $O(\log N)$ bits are required for storing the height.

In case of union-by-size, the depth of any node is never more than $\log N$ (proved in lecture 9). Thus, the height of the tree is upper bounded by $\log N$. $O(\log \log N)$ bits are required for storing the height.

Using union-by-height, the height of any tree is also $O(\log N)$. $O(\log \log N)$ bits are required for storing the height.

Problem 2: 6 pts Let A be the adjacency matrix of a directed graph. Consider the matrix A^2. Is there any relation between the entry $A^2[i,j]$ to the number of paths between vertices i,j in the original graphs? Justify your answer. Answer the same question about $A^n[i,j]$.

Each entry $A^2[i,j]$ corresponds to the number of paths between i,j of length exactly 2. The reason is the following:

Assume that the graph has n vertices. The matrix multiplication A^2 will produce an array with the following entries:

$$A^2[i,j] = A[i,1] * A[1,j] + A[i,2] * A[2,j] + \ldots + A[i,n] * A[n,j]$$

Each subterm $A[i,k] * A[k,j]$ is 1 iff both $A[i,k] = 1$ and $A[k,j] = 1$. This means that there is an edge from i to k and from k to j. So, we can derive that there is a path from i to j of length 2. Thus, the total number of subterms which constitute a path of length 2 will be assigned to $A^2[i,j]$.

It is easy to prove by induction that $A^n[i,j]$ corresponds to the number of paths between i,j of length exactly n. We have already proved the base case for $n=2$ (it also holds trivially for $n=1$). Assume that it holds for $k-1$, and prove that it also holds for k, where $A^k = A^{k-1} * A$.

Problem 3: 6 pts Consider the graph representation of a complete binary tree. How many edges can a complete binary tree of n nodes have? How will you represent it (adjacency list or adjacency matrix)? Give the storage complexity of your representation.

Solution:

Every tree of n nodes is a connected graph with $|E| = n-1$ edges. The complete binary tree is a special class of tree, thus it also has $n-1$ edges. Since $|E| = n-1 = \Theta(n)$, the complete binary tree is a sparse graph. Thus, it should be represented as an adjacency list.

If G is an undirected graph, the sum of the lengths of all the adjacency lists is $2|E|$, since if (u,v) is an undirected edge, then u appears in v's adjacency list and vice versa.

Thus, the storage complexity for the complete binary tree is $2|E| = 2(n-1) = \Theta(n)$.

Problem 4: 6 pts Consider an adjacency list representation of a digraph. Give algorithms to compute the in-degree and out-degree of a vertex using the adjacency list representation. Analyze their complexities.

Solution:

```
Out-Degree(G,v)
Input: Digraph G=(V,E) and vertex v
Output: out(v)
{
out(v)=0;

for w in Adj[v] /* traverse the linked list Adj(v) */
   out(v)++;
}

In-Degree(G,x)
Input: Digraph G=(V,E) and vertex x
Output: in(x)
{
in(x)=0;

for v in V
 for w in Adj[v] /* traverse the linked list Adj(v) */
    if (w==x)
        in(v)++;
}
```

Since G is a directed graph, the sum of the lengths of all the adjacency lists is $|E|$. Thus, both algorithms are computed in $\Theta(|E|)$.

Problem 5: 6 pts Let A be the adjacency matrix of a graph G. Let G^T be a graph with adjacency matrix A^T. (A^T is transpose of a matrix A. Transpose of a matrix A, A^T is related to A as follows: $A^T[i,j] = A[j,i]$.) Do you see any relation between G and G^T? Answer the question for a digraph G.

Solution:

In case of an undirected graph G, (i,j) and (j,i) represent the same edge. Thus, the adjacency matrix A of an undirected graph is its own transpose: $A = A^T$. Consequently, G and G^T represent the same graph.

In case of a digraph G, G^T is G with all its edges reversed.