论题 2-16 作业

姓名：陈劭源　　学号：161240004

1 [TC] Problem 22.1-3

For adjacency-list representation:

```
TRANSPOSE(G)
1  let Adj[1..|V(G)|] be a new array of lists
2  for i = 1 to |V(G)|
3      for each e in G.Adj[i]
4          Adj[e].insert(i)
5  return graph (V(G),Adj)
```

For adjacency-matrix representation:

```
TRANSPOSE(G)
1  let A[1..|V(G)|,1..|V(G)|] be a new array
2  for i = 1 to |V(G)|
3      for j = 1 to |V(G)|
4          A(i,j) = G.A(j,i)
5  return graph (V(G),A)
```

2 [TC] Problem 22.1-8

If we use hash table with collision resolution by chaining, the expected time to determine whether an edge is in the graph is $O(1+\alpha)$, where α is the load factor. (We will not adopt open addressing, because it is no better than adjacency-matrix, i.e. direct-address tables)

The disadvantage of this scheme is that, we still need a great number of memory space, even if the graph is very sparse.

Instead of a hash table, we can use binary search tree as array entry $Adj[u]$, containing the vertices v for which $(u,v) \in E$. This alternative requires $O(\log n)$ time for determining whether an edge is in the graph, where n is the out-degree of u, usually worse than hash table.