Recap of this week’s lectures:

- DFS: topological sort, cycle detection, biconnected and strongly connected components

**SCC Problem**

We are given a directed graph $G$ with $N$ nodes and $M$ edges. Each node $v$ has a treasure chest containing $c_v > 0$ value. You are allowed to start and end anywhere in the graph, and are allowed to visit each node and traverse each edge multiple times. At any point in time, you can loot the treasure chest of your current node, but each chest can be looted at most once! Design an $O(N + M)$ algorithm to calculate the maximum value you can obtain.
Ear Decomposition
Throughout these notes let $G = (V, E)$ be a simple, no multiple edges, undirected graph with at least three vertices. Let $n$ be the number of vertices and $m$ the number of edges.

In lecture we said that $G$ is bi-connected if it does not contain any articulation points. This definition was carefully chosen so that $G$ would have a partition of the edge into bi-connected components. Thus a single edge is bi-connected.

**Lemma 1** $G$ is connected and bi-connected and contains at least three vertices if for every pair of vertices there are two vertex-disjoint (except at the endpoints) paths between them.

**Definition 1** An ear decomposition $ED$ of $G$ is a partition of the edges of $G$ into simple paths $P_1, \ldots, P_k$.

1. $P_1$ is a simple cycle.
2. $P_2, \ldots, P_k$ are open paths, their endpoints are distinct.
3. The end points of the path $P_{i+1}$, its attachments, belong to the vertices of $P_1 \cup \cdots \cup P_i$ for $1 \leq i < k$.

The following figure shows a graph and an ear decomposition of it. The cycle $P_1$ is shown in blue, the first path $P_2$ is shown in red, the second path $P_3$ is shown in green, and the last path (just an edge) $P_4$ is shown in purple.

![Graph with ear decomposition]

Each $P_i$ is called an ear.

**Lemma 2** The number of ears of a $ED$ is $m - n + 1$. 
Lemma 3  The number of back-edges of every DFS is \( m - n + 1 \).
Programming Problem Hints:

Your algorithm must make a new ear from each back-edge (why?). Thus the main subgoal of your algorithm is to assign each tree edge to a back-edge/ear.

Recall that if \( T \) is a spanning tree and \( e \) is a non-tree edge then there exists a unique cycle consisting of \( e \) and edges from \( T \). We denote this cycle by \( C_e \). All edges in the ear belonging to back edge \( e \) must come from \( C_e \).

The next issue is that if there are several cycles all using a given tree edge, we need to decide which backedge/ear this tree edge belongs to. What criteria should we use? Consider the case when the DFS tree is a line.