• **DISCLAIMER:** These notes are not necessarily an accurate representation of what I said during the class. They are mostly what I intend to say, and have not been carefully edited.

• **Tree Contraction**
  - Given a weighted, bounded degree tree, compute the depth of each node
  - Generalization of list ranking: line is a tree.
  - $O(n)$ work sequential algorithm: DFS / BFS.
  - Goal: $O(n)$ work, $O(\log n)$ depth.

• **Operations**
  - Contract degree 2 vertices.
    * Nodes with 1 child, pick $O$ or $E$, randomly.
    * Node with 0 children (leaves), pick $E$.
    * $E$ contract into $O$.
  - Problem: everything can have degree 2 parents.
  - Fix: degree 2 vertices pick $L$ or $R$, randomly.
  - Progress:
    * Leaf: $1/2$ chance that parent picks it.
      - Parent with 1 child: $1/2$ chance of picking $O$.
      - Parent with 2 children: $1/2$ chance of pointing to ‘right’ child.
    * Degree 1 node: need to also pick $E$, $1/2 \times 1/2 = 1/4$.

• **Convergence:**
  - Claim: tree becomes a single vertex w.h.p. after $O(\log n)$ rounds.
  - Let $v_0$, $v_1$, and $v_2$ be the number of nodes with 0, 1, and 2 children.
  - Expected progress: $v_0/2 + v_1/4$.
  - Total number of vertices: $n = v_0 + v_1 + v_2$.
  - Total number of edges: $n - 1 = v_1 + 2v_2$. 

1
Subtract: \( v_0 = v_2 + 1 \), substituting:

\[
v_0/2 + v_1/4 \geq v_0/4 + v_1/4 + v_2/4 \geq n/4
\]

- Markov’s inequality: constant probability for a constant factor size reduction.
- \( O(\log n) \) rounds w.h.p.: see problem set.
- (general trick) High degree vertices: turn into a balanced binary tree.

- **Application:** binary expression evaluation
  - Tree corresponds to an arithmetic expression involving + and \( \times \).
  - Contracted nodes: \( x \rightarrow ax + b \), more intricate intermediate states.

- **Connection to data structures**
  - Can view intermediate trees as a hierarchy of nodes.
  - Balanced tree built on the original tree.
  - Local updates propagate up \( O(\log n) \) levels.
  - \( O(\log n) \) time data structures that supports:
    * Modify length of edge.
    * Query node’s distance to root.
  - Also works for expression evaluation.
  - Connection between low depth and efficient data structures can be formalized.

- **Dynamic trees / Top trees**
  - Supports the following operations in \( O(\log n) \) time:
    * Topological changes.
    * Updates to nodes (also paths with a bit more work).
    * sum / min of paths
    * Formally, any associate function suffices, aka. \( f(f(f(a,b),c),d) = f(f(a,b),f(c,d)) \).
  - Constructive view: every tree is a balanced binary tree (height \( O(\log n) \)) of paths (representable as \( O(\log n) \) depth trees).
  - Applications
    * Incremental minimum spanning trees.
    * Fully dynamic graph algorithms: [http://dl.acm.org/citation.cfm?id=502095](http://dl.acm.org/citation.cfm?id=502095).