1. Goal: more streamlined dynamic tree data structure
   (a) Recall: splay trees, can move any node to root.
   (b) Formal statement of splay theorem: (should I prove this?)
      i. $S_u$: size of u’s subtree
      ii. Cost of splaying $u$ to root of tree at $v$: $1 + O(\log(S_v/S_u)) = 1 + O(\log(S_v)) - O(\log(S_u))$.
   (c) Recall: heavy-light decomposition: each node has one preferred child. Store path formed by preferred edges in BBSFs.
   (d) Idea: store each path using a splay tree.
   (e) Trickiness: there are two kinds of trees going on here now. One ‘real’ tree, one ‘virtual’ tree, which is BBST representation of paths.
   (f) Left in virtual trees means higher up in the real tree.
   (g) Next idea: each time we ‘touch’ a node, make its entire path to root in the real tree preferred edges.
   (h) Details
      i. Nodes in a virtual tree tracks the parent (non-preferred edge) in real tree.
      ii. To fix path of $u$, let its parent in real tree be $v$.
      iii. Splay $u$ and $v$ (in different trees).
      iv. Detach right child of $u$, attach $v$ to it as right child.
   (i) Total cost per splay: let top of each preferred path be $v_0, v_1 \ldots v_l$.
   (j) $\sum_i 1 + O(\log(S_{v_i})) - O(\log(S_{v_{i-1}})) \leq l + O(\log n)$.
   (k) Claim: number of changes of preferred child is $O(\log n)$.
   (l) ‘heavy’ edge in real tree: edge to subtree of larger size.
   (m) Each path can encounter at most $O(\log n)$ non heavy edges.
   (n) Potential: number of preferred children that are not heavy edges.
   (o) Each access can increase potential by at most $O(\log n)$.
   (p) Total length of $l$ over $n$ accesses: $O(n \log n)$.

2. Incremental Minimum Spanning TRee
   (a) Need to support: maximum weight edge along a tree path.
   (b) Given two nodes, ‘touching’ them in order makes path to LCA contain two paths.
   (c) Query for min of a subtree of the virtual tree, done by storing the min of each subtree in virtual tree (corresponding to a chunk of a path in real tree).