Link-cut Trees

Using link-cut trees we want to maintain a forest of rooted trees whose each node has an arbitrary number of unordered child nodes. The data structure has to support the following operations in O(lg n) time:

Used in many applications such as Network Flow and Dynamic Connectivity problems

Operations

• **make tree()** – Returns a new vertex in a singleton tree. This operation allows us to add elements and later manipulate them.

• **link(v,w)** – Makes vertex v a new child of vertex w, i.e. adds an edge (v, w). In order for the representation to remain valid this operation assumes that v is the root of its tree and that v and w are nodes of distinct trees.

• **cut(v)** – Deletes the edge between vertex v and its parent, parent(v) where v is not the root.

• **find root(v)** – Returns the root of the tree that vertex v is a node of. This operation is interesting because path to root can be very long. The operation can be used to determine if two nodes u and v are connected.

• **path aggregate(v)** – Returns an aggregate, such as max/min/sum, of the weights of the edges on the path from the root of the tree to node v. It is also possible to augment the data structure to return many kinds of statistics about the path.

Definition of Link-Cut Trees

We say a vertex has been accessed if was passed to any of the operations from above as an argument. We call the abstract trees that the data structure represents **represented trees**. We are not allowed to change the represented tree and it can be unbalanced. The represented tree is split into paths (in the data structure representation).

The **preferred child** of node v is equal to its i-th child if the last access within v's subtree was in the i-th subtree and it is equal to null if the last access within v's subtree was to v itself or if there were no accesses to v's subtree at all. **A preferred edge** is an edge between a preferred child and its parent**. A preferred path** is a maximal continuous path of preferred edges in a tree, or a single node if there is no preferred edges incident on it. Thus preferred paths partition the nodes of the represented tree.

Link-Cut Trees represent each tree T in the forest as a tree of **auxiliary trees**, one auxiliary tree for each preferred path in T. Auxiliary trees are splay trees with each node keyed by its depth in its represented tree. Thus for each node v in its auxiliary tree all the elements in its left subtree are higher(closer to the root) than v in v's represented tree and all the elements in its right subtree are lower. Auxiliary trees are joined together using **path-parent pointers**. There is one path-parent pointer per auxiliary tree and it is stored in the root of the auxiliary tree. It points to the node that is the parent(in the represented tree) of the topmost node in the preferred

path associated with the auxiliary tree. We cannot store path-to-child pointers because there can be many children. Including auxiliary trees with the path-parent pointers as edges, we have a representation of a represented tree as a tree of auxiliary trees which potentially can have a very high degree.

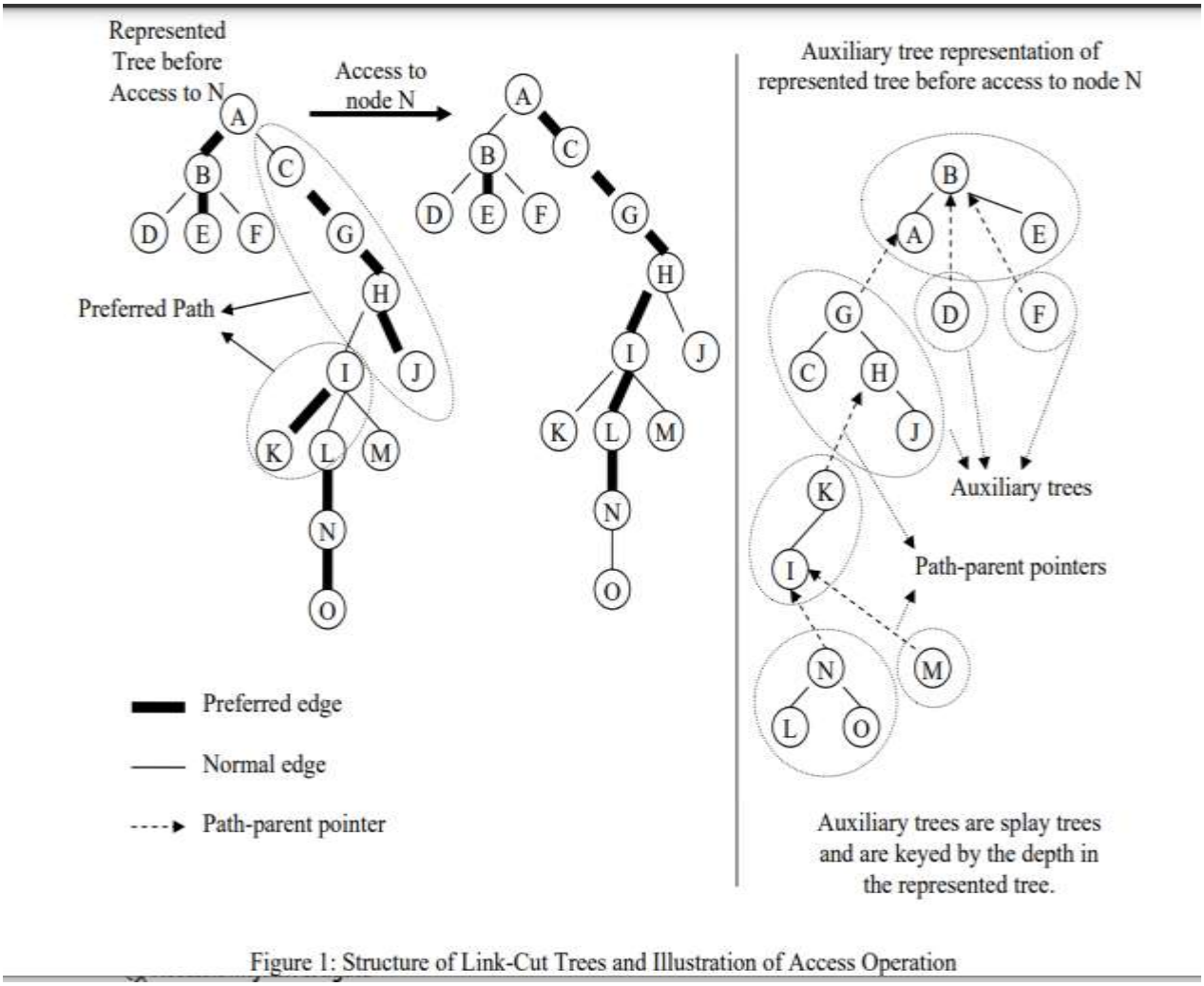

Figure 1: Structure of Link-Cut Trees and Illustration of Access Operation