6.851 Advanced Data Structures (Spring'07)

Prof. Erik Demaine TA: Oren Weimann

Problem 3 – Solution

Dynamic connectivity with key values. The key idea is to preform the added operations only on the ET forest of $F_{\lg n}$. We assign a key to each node in an ET-tree. The value of this key is set to ∞ for all nodes but those that represents a first occurrence in the Euler tour. Every node in the ET tree also stores the smallest value associated with any node within the ET-subtree rooted at it.

find-min(v) and *set-key(v,x)* are both done by traversing the path from v to its root in the ET forest of $F_{\lg n}$ (the latter also fixes the minimum value fields along the path). Also, during the BST splitting (that is done when we link or cut ET-trees) we fix the minimum value fields appropriately. If we use ET-trees with branching factor of 2 we get that set-key takes $O(\lg n)$ time as the ET-trees are balanced. However, since we use ET-trees with branching factor of 2 we don't have the $O(\lg n / \lg \lg n)$ time for findroot (but rather $O(\lg n)$).

An alternative solution (also accepted) is to use ET-trees with branching factor of $\lg n$. This way, findroot takes $O(\lg n / \lg \lg n)$ but set-key takes $O(\lg^2 n / \lg \lg n)$ which is worse than $O(\lg n)$.

Dynamic connectivity with path queries. The main idea is "the difference between knowing the path and walking the path". We can check if a path exists by doing *findroot* on both v and w in the ET-tree of $F_{\lg n}$ (in $O(\lg n / \lg \lg n)$ time). If indeed they are connected, we trace a path by tracing parent pointers of v and w in $F_{\lg n}$. Notice that the parent pointers represent $F_{\lg n}$ and not its ET tree. Such parent pointers can be easily updated when operations are done on the ET tree.

An alternative solution is to find the parent pointers using the ET-tree (rather than explicitly storing them). To find the parent pointer of v, find it's last occurrence in the ET-tree (in O(1) time). v's parent is the next node in the Euler tour (also can be found in O(1) time).