cs201c: Tutorial 3
Topics: B-trees, Scapegoat trees, Splay trees, and Treaps

Join operation. The input consists of two BSTs T1 and T2. All keys in tree T1 are less than all keys in tree T2. Construct a single BST T consisting of the union of all keys in T1 and T2. (The original trees are destroyed in the process.)

Split operation. The input consists of a single BST T and a key k. Break tree T into two BSTs T1 and T2 such that (i) the union of all keys in T1 and T2 is exactly the set of keys in T, (ii) every key in tree T1 is <=k and every key in tree T2 is >k. (The original tree T is destroyed in the process.)

A. B-trees

- Design top-down (one pass) insert and delete operations for a B-tree of order t (t>=2).
- Design a join operation on B-tree (of any fixed order t), that takes O(log_{t}(n)) disk accesses. [n is the total number of nodes in T1 and T2.]
- Design a split operation for B-trees (of any fixed order t) that takes O(log_{t}(n)) disk accesses. (As a first step, you may first design a split operation which takes O([log_{t}(n)]^2) disk accesses.) [n is the number of nodes in T.]

B. Scapegoat trees

- Give an example of a scapegoat tree T on n nodes, such that a suitable insert operation on tree T leads to its root being the scapegoat.
- Design delete operation in a scapegoat tree. Show that the amortized cost of your delete operation is O(log_2(n)) by using (a suitable modification of) the potential function for scapegoat trees discussed in class. (Here n is the number of nodes/keys in the scapegoat tree.)

  (Hint. Maintain two integers m and n. n is the number of keys currently in the scapegoat tree, and m is the maximum number of keys at any point in the tree since the last delete operation. Clearly, m>=n.

  A scapegoat tree is now defined as any tree with height at most log_{1.5}(m).

  During insert, increase n by 1. Also, increase m by 1, if m=n.
  During delete decrease n by 1. Fully rebalance the root of the tree if 2n<m and then set m=n.)

C. Splay trees

- Show that amortized cost of insert and delete operations in splay trees is O(log_2(n)).

- Design a join operation for splay trees with amortized cost O(log_2(n)), where n is the total number of keys in T1 union T2.
- Design a split operation for splay trees with amortized cost O(log_2(n)) [n is the number of keys in the original splay tree T].

D. Treaps.

- Recall the letter posting problem discussed in class, where n letters are randomly distributed among n envelopes.

  We say that a letter pair (i, j) [here 1<=i<j<=n] is *interchanged* iff letter i goes to envelope j and letter j goes to envelope i.

  Let Y1 be a random variable which is equal to the total number interchanged letter pairs. Derive the expected value E[Y1].

- Design a join operation for treaps with expected cost O(log_2(n)). Here, n is the total number of keys in T1 union T2.
- Design a split operation for treaps with expected cost O(log_2(n)) [n is the number of keys in original treap T].

- [Based on Practice Lab 2.] Do average-case analysis of insertion sort, when the permutation *tau* to be sorted [see Lab Question 3, parts (ii) and (iii) in Practice Lab 2] is obtained by the file delivery process with the end-to-end delay following the Pareto distribution with alpha=1 and beta=1. What about alpha=2 and beta=1?

  (Hint. Define indicator random variables Z_{i,j} (1<=i<j<=N). Z_{i,j} is equal to 1 if and only if packet Pi arrives after packet Pj at computer B. Express your answer in terms of these indicator random variables and then apply linearity of expectation.)