1 Skiplists

Recall that a skiplist stores elements in a sequence of smaller and smaller lists $L_0, \ldots, L_k$. L_i is obtained from L_{i-1} by tossing a coin for each element in L_{i-1} and including the element in L_i if that coin comes up heads.

1. Draw an example of a skiplist, select a few elements and show the search paths for these elements.

2. Explain how the reverse search path is related to the following experiment: Toss a coin repeatedly until the first time the coin comes up heads

3. If there are n elements in L_0, what is the expected number of elements in L_1? What about in L_i?

4. If there are n elements in L_0, give an upper bound on the expected length of the search path for any particular element.

5. Explain, briefly, how a skiplist can be used to implement the `SortedSet` interface. What are the running times of operations `add(x)`, `remove(x)`, `contains(x)`?

6. Explain, briefly, how a skiplist can be used to implement the `List` interface. What are the running times of the operations `add(i,x)`, `remove(i)`, `get(i,x)` and `set(i,x)`

2 Big-Oh Notation

1. What is $O(n)$? What kind of object is it?

2. What does the statement $2n + 3 = O(n^2)$ mean?

3. Recall the basic big-O hierarchy: for any constants $a, b > 0$:

$$O(a) \subset O(\log n) \subset O(n^b) \subset O(c^n)$$

   Where do the functions $2n^2$, $100n \log n$, and $(\log n)^3$ fit into this hierarchy?

3 Convex Hulls

1. Draw a set of points and illustrate the operation of Graham's Algorithm for Computing the upper hull of your point set

4 Binary Trees

1. Draw a good sized binary tree
2. Label the nodes of your drawing with their depth
3. Label the nodes by the order they are processed in a preorder traversal
4. Label the nodes by the order they are processed in a inorder traversal
5. Label the nodes by the order they are processed in a postorder traversal
6. Label the nodes of your tree in such a way that you can tell if a node u is an ancestor of a node w just by looking at their labels
7. Label the nodes of your tree with the sizes of their subtrees
8. What kind of traversal would you do to compute the sizes of all subtrees in a tree?

5 Binary Search Trees

1. Consider an array containing the integers 0,...,15 in sorted order. Illustrate the operation of binary search on a few (a) integer values and a few (b) non-integer values
2. Draw a binary search tree containing 0,...,15
3. Show the search path for a value x in the tree and a value x' not in the tree
4. Insert some value x' into the tree
5. Delete some value x from an internal node of the your tree (preferably one having two children)
6. Choose a permutation of $\pi_0, \ldots, \pi_{15}$ of 0,...,15 and draw the binary search tree that results from inserting the elements of your permutation in your order. Are there other permutations that could have given the same search tree?
7. Explain the relationship between quicksort and random binary search trees

6 Treaps

1. Define the heap property for priorities in a binary tree
2. Pick some random numbers $p_0, \ldots, p_{15}$, and draw the treap that contains $0, \ldots, 15$ where p_i gives the
3. Explain the relationship between random binary search trees and treaps
4. Explain the relationship between insertion and deletion in a treap

7 Scapegoat Trees

Recall that a *scapegoat node* is a node `v=u.parent` where `2*size(u) > 3*size(u.parent)`

1. In a scapegoat tree, is `size(u)` $\leq$ `size(u.parent)` for every non-root node `u`?
2. Draw an example of a tree that looks surprisingly unbalanced but is still a valid scapegoat tree
3. Explain why, in a scapegoat tree, we keep two separate counters `n` and `q` that—sort of—keep track of the number of nodes?
4. If `v` is a scapegoat node (for example, because `3*size(v.left) > 2*size(v)`) then explain how many operations (insertions/deletions) have affected `v`'s subtree since the last time the subtree containing `v` was rebuilt

8 Heaps

These questions are about complete binary heaps represented using the Eytzinger Method.

1. Draw an example of a complete binary heap with key values
2. Illustrate how your example's values map into an array using the Eytzinger Method.
3. Consider a binary heap represented using the Eytzinger Method. Give the formulas for a the parent, left child, and right child of the value stored at position `i`.
4. When we perform a DeleteMin on a heap, where do we get the value to replace the root and what do with it
5. Explain, or illustrate, the insertion algorithm for a heap
6. Explain the operation of HeapSort

9 Randomized Meldable Heaps

1. Draw two examples of heaps (not necessarily complete) and show how to meld (merge) them.

10 Sorting

1. Explain the relative advantages of MergeSort, HeapSort, and Quicksort. Things to keep in mind are (a) the number of comparisons performed, (b) the amount of extra memory required by the algorithms, and (c) whether their running time is guaranteed or only probabilistic.

11 Plane Sweep

1. Draw a set of line segments in the plane, some of which cross each other
2. List the endpoint events in the Bentley-Ottman segment intersection algorithm for your set of points
3. List the intersection events in the Bentley-Ottman segment intersection algorithm for your set of points. For each event, determine which previous (intersection or endpoint) event generated it.

12 2-4 and Red-Black Trees

1. Draw an interesting example of a 2-4 tree
2. Explain what happens in 2-4 tree when an insertion causes a node to have more than 4 children
3. Explain what happens in a 2-4 tree when a deletion causes a node to have only one child
4. Draw a red-black tree that corresponds to your 2-4 tree
5. Explain the relationship between the red-black tree properties and the 2-4 tree properties