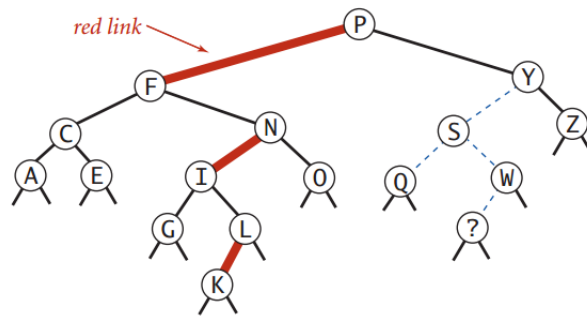


## Week 5 Flipped Activities

### 1. LLRB's

Consider the following left-leaning red-black BST. Some of the colors and key values are suppressed.



- (a) Which of the keys below could be the one labeled with a question mark? Circle all possibilities. *ABCDEFGHIJKLMNOPQRSTUVWXYZ*
- (b) For each link below, select its possible color(s) from red, black or either red or black.
  - i. link between W and S
  - ii. link between ? and W
  - iii. link between S and Y
  - iv. link between Q and S
- (c) How many left rotation, right rotation, and color flip operations would be used to insert each key below into the original red-black BST above?

	H	D	B	J
<code>rotateLeft()</code>	1			
<code>rotateRight()</code>	0			
<code>flipColors()</code>	0			

## 2. Comparing two arrays of points.

Given two arrays  $a[]$  and  $b[]$ , each containing  $N$  distinct points in the plane, design two algorithms (with the performance requirements specified below) to determine whether the two arrays contains precisely the same set of points (but possibly in a different order).

For each algorithm, give a crisp and concise English description of your algorithm. Your answer will be graded on correctness, efficiency, and clarity.

- (a) Design an algorithm for the problem whose running time is linearithmic in the worst case and uses at most constant extra space.
- (b) Design an algorithm for the problem whose running time is linear under reasonable assumptions and uses at most linear extra space. Be sure to state any assumptions that you make.

## 3. Hashing with Linear Probing

Suppose that the following keys are inserted into an initially empty linear-probing hash table, but not necessarily in the order given

<i>key</i>	<i>hash</i>
A	1
D	5
L	6
M	0
N	1
S	6
X	4

and it result in the following hash table:

0	1	2	3	4	5	6
S	M	N	A	X	D	L

Assuming that the initial size of the hash table was 7 and that it did not grow or shrink, circle all possible keys that could have been the last key inserted. *ADLMNSX*

#### 4. Miscellaneous Questions

For the following problems, state whether it is POSSIBLE, IMPOSSIBLE or OPEN

- (a) Given any array of  $N$  distinct integers, determine whether there are three integers that sum to exactly zero in time proportional to  $N^{1.5}$ .
- (b) Given any array of  $N$  distinct integers, determine whether there are three integers that sum to exactly zero in time proportional to  $N^2$ .
- (c) Implement a FIFO queue with a constant amount of memory plus two LIFO stacks, so that each queue operation uses a constant amortized number of stack operations.
- (d) Given any left-leaning red-black BST containing  $N$  keys, find the largest key less than or equal to a given key in logarithmic time.
- (e) Design a priority queue implementation that performs insert, max, and delete-max in  $(1/3) * \log_2 N$ , where  $N$  is the number of comparable keys in the data structure.
- (f) Given any array of  $N$  keys containing three distinct values, sort it in time proportional to  $N$  and using only a constant amount of extra space.
- (g) Design a practical, in-place, stable, sorting algorithm that guarantees to sort any array of  $N$  comparable keys in at most  $N \lg N$  compares.
- (h) Find the  $k$ th smallest key in a left-leaning red-black BST in logarithmic time in the worst case.

## 5. Problem Identification

You are applying for a job at a new software technology company. Your interviewer asks you to identify the following tasks as either possible (with algorithms and data structures learned in this course), impossible, or an open research problem. You may use each letter once, more than once, or not at all.

- (a) Implement a union-find data type so that all operations (except construction) take logarithmic time in the worst case. Find two strings in Java that are not equal but have the same hashCode()
- (b) Design a compare-based algorithm to merge two sorted arrays, each of length  $N/2$ , into a sorted array of length  $N$  that makes  $(1/2) * N$  compares in the worst case.
- (c) Given a binary heap on  $N$  distinct keys, build a BST containing those keys using  $17N$  compares in the worst case.
- (d) Given a binary search tree (not necessarily balanced) on  $N$  distinct keys, build a binary heap containing those  $N$  keys using  $17N$  compares in the worst case.
- (e) Design a stable compare-based sorting algorithm that sorts any array of  $N$  comparable keys using  $N \log_2 N$  compares in the worst case.
- (f) Given a sorted array of  $N$  keys (not necessarily distinct), find the number of keys equal to a given query key using  $2 \log_2 N$  compares in the worst case.