Data Structures

Fall 2019, Final Exam. (Solutions)

1. (9 pts) Consider *Insertion-Sort*, *Quick-sort* (using the last element as the pivot), and *Merge-Sort*. What will be the worst case asymptotic upper bound on the running time for each of the algorithms if you know additionally that (i) the input is already sorted? (ii) the input is reversely sorted? (iii) the input is a random sequence? Give your answer in $O(...)$. In your answer, for each of (i), (ii), (iii), give the worst case upper bounds for *Insertion-Sort*, *Quick-sort* and *Merge-Sort*.

Sol.

	(i)	(ii)	(iii)
Insertion-sort	$O(n)$	$O(n^2)$	$O(n^2)$
Quick-sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Merge-sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$

2. (11 pts) Consider a hash table consisting of $M = 11$ slots, and suppose nonnegative integer key values are hashed into the table using the primary hash function $h_1()$:

```
int h1 (int key) {
    int x = (key + 7) × (key + 7);
    x = x / 16;
    x = x + key;
    x = x % 11 ;
    return x }
```

The integer key values listed below are to be inserted, in the order given $\mathbf{43}, \mathbf{23}, \mathbf{1}, \mathbf{0}, \mathbf{15}, \mathbf{31}, \mathbf{4}, \mathbf{7}, \mathbf{11}, \mathbf{3}$.

(a) Complete the following table for the home slot (the slot to which the key hashes using h_1, before any probing).

Sol.

key	43	23	1	0	15	31	4	7	11	3
Home slot	1	2	5	3	1	$\boxed{0}$	$\boxed{0}$	$\boxed{8}$	$\boxed{9}$	$\boxed{9}$

(b) Suppose that collisions are resolved by using double hashing (i.e., $h(key) = (h_1(key) + i \times h_2(key))\ mod\ 11$, $i = 0, 1, ...$), with the secondary hash function $h_2(key) = Reverse(key)$, which reverses the digits of the key and returns that value; for example, $Reverse(7823) = 3287$.

Complete the contents of the final hash table.

Sol.

slot	0	1	2	3	4	5	6	7	8	9	10
contents	$\boxed{31}$	43	23	0	$\boxed{4}$	1		$\boxed{7}$	$\boxed{15}$	$\boxed{11}$	$\boxed{3}$

3. (10 pts) In the potential method for the amortized analysis of splay trees, we define the *rank* of a node q in a tree T as $rank(q) = \lfloor \log_2 size(q) \rfloor$, and $size(q)$ equals the number of nodes (including q) in the subtree rooted at q. It is also known that when we splay a node x in a splay tree T, the amortized cost of the splay operation is $\leq 3(rank(root) - rank(x)) + 1$.

(a) (5 pts) Suppose that at some point in time the amortized const of a splay at node x is 25. What is the smallest possible number of nodes in the tree? Why?
Sol. $\boxed{256}$. 3(rank(root)-rank(x))=25, so rank(root) is at least 8 and the number of nodes is at least $2^8 = 256$.

(b) (5 pts) What is the change to the value of $rank(x)$ (i.e., the rank of x after the splay minus the rank of x before the splay) as a result of this splay? Why?
Sol. $\boxed{8}$. After the splay, x is the root of the tree, so it has the same rank that r had originally. So, the rank(x) increases by 8.

4. (15 pts) Consider the following data structure operations to maintain a set S of n numbers:

   - $i(x)$: insert a new element x to S
   - $c(x, k)$: change an element x's key to a given number k;
   - $m()$: return the *medium* of S, i.e., the $\lfloor |S|/2 \rfloor$-th smallest element in S. (Note, $|S|$ is the size of S, and, e.g., $\lfloor \frac{5}{2} \rfloor$=2.)

   (a) (9 pts) For the data structure S, you may use an extra key $-\infty$ (resp., ∞) which is smaller (resp., greater) than all the keys of A. For instance, suppose $A[0..3] = [5, 3, 6, 4]$ and initially S is empty. For the following sequence, the contents of S and B after each operation would be:

op	$i(A[0])$	$i(A[2])$	$B[0] = m()$	$c(B[0], \infty)$
S	$\{5\}$	$\{5, 6\}$	$\{5, 6\}$	$\{\infty, 6\}$
B	[-, -, -, -]	[-, -, -, -]	[5, -, -, -]	[5, -, -, -]

   where "-" means "don't care". Write a sequence of operations to output a sorted array $B[0..3]$ (in increasing order) containing the same keys as A. To this end, complete the following table:

   Sol.

$i(-\infty)$, $i(-\infty)$, $B[0] = m()$	$i(\infty)$, $i(\infty)$, $B[1] = m()$	$i(\infty)$, $i(\infty)$, $B[2] = m()$	$i(\infty)$, $i(\infty)$, $B[3] = m()$
$\{-\infty, -\infty, 3, 4, 5, 6\}$	$\{-\infty, -\infty, 3, 4, 5, 6, \infty, \infty\}$	$\{(2 \times -\infty), 3, 4, 5, 6, (4 \times \infty)\}$	$\{(2 \times -\infty), 3, 4, 5, 6, (6 \times \infty)\}$
[3, -, -, -]	[3, 4, -, -]	[3, 4, 5, -]	[3, 4, 5, 6]

   (b) (3 pts) Suppose we consider $A[0..n-1]$, i.e., an array of n elements, how many $i(), d(), m()$ operations together are needed to output a sorted array $B[0..n-1]$? Write your answers in $O()$.

   Sol. $\boxed{O(n)}$.

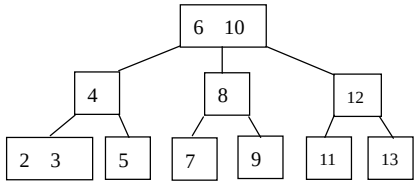
   (c) (3 pts) Suppose the implementation of $i(), d(), m()$ involves key comparisons (i.e., $a < b$, $a > b$, $a = b$), and we know that any comparison-based sorting requires $\Omega(n \log n)$ time. Explain why no data structure for S can achieve $O(\sqrt{\log n})$ amortized time for $i()$, $d()$, and $m()$ simultaneously.

   Sol. $\boxed{\text{Otherwise, comparison-sorting would have been done in } O(n\sqrt{\log n}) < O(n \log n).}$

5. (12 pts) Consider the following AA-tree in Figure 1.

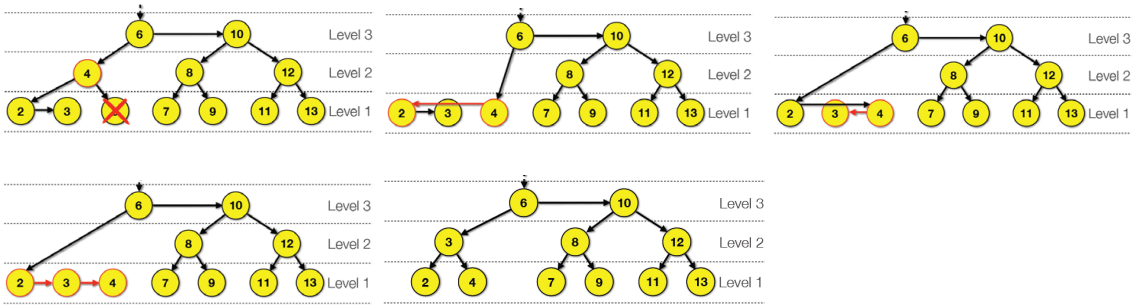
   (a) (5 pts) Draw the corresponding 2-3 tree.

   Sol.



   (b) (7 pts) Delete 5 in the AA-tree and then rebalance the tree. Show your intermediate steps.

   Sol.



6. (12 pts) Recall that the priority queue ADT plays a key role in Prim's algorithm for finding the minimum spanning tree (MST) of a graph. Figure 2 shows an incomplete representation of an intermediate state in the execution of Prim's algorithm. Figure 2(left) is the original graph with the partial solution annotated by dark edges (i.e., the partial minimal spanning tree containing the edges (b, g) and (g, e)). Figure 2(right) represents the underlying priority queue implemented using a binary heap (with the ∞ value keys omitted).

   (1). (4 pts) In addition to the priority queue, in the implementation we also use any array *cheap* to record the edge with the smallest weight connecting to the partial MST. For instance, for the intermediate step above, $cheap(a) = (a, b)$ as (a, b) has the minimum weight (i.e., 5) connecting to the partial MST consisting of edges (b, g) and (g, e). The $cheap(b) =$ " $-$ " as b is already in the partial MST. In this question, complete the *cheap* array.

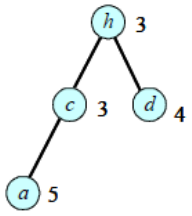
   Sol.

	a	b	c	d	e	f	g	h
cheap	(a, b)	–	(b, c)	(d, e)	–	(e, f)	–	(b, h)

   (2). (8 pts (2, 3, 3)) Show how the next step of the prim's algorithm modifies the data structure. To this end, you need to (1) give the next edge added to the partial MST, (2) draw the new binary heap (after all the updates are done), and (3) give the new *cheap* array.

   Sol.

   i. (e, f)



   ii.

   iii.

	a	b	c	d	e	f	g	h
cheap	(a, b)	–	**(c, f)**	(d, e)	–	–	–	(b, h)

7. (12 pts) In Union-Find data structures, we often use an array for the parent-link representation. Take array B in the following table for instance. The parent of node 0 is node 4, as $a[0] = 4$, and 1 is a root as $a[1] = 1$. Determine whether each of the four arrays could possibly occur during the execution of a union-find data structure with *union by height*. Why?

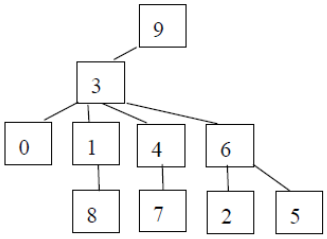
```
                0  1  2  3  4  5  6  7  8  9
               ---------------------------
A.   a[i]:     8  0  4  0  0  4  0  4  2  0

B.   a[i]:     4  1  8  2  1  5  1  1  4  5

C.   a[i]:     3  3  6  9  3  6  3  4  1  9

D.   a[i]:     2  1  1  1  1  1  1  2  1  7
```

   Sol.

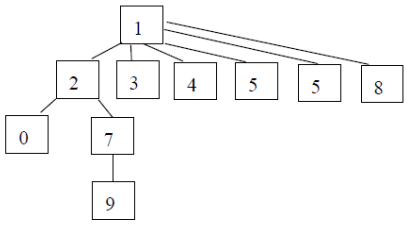
   (a) No. $8 \to 2 \to 4 \to 0 \to 8$ is a cycle.

   (b) No. The height is $4 > \log 10$.

(c) No. Consider nodes 3 and its parent 9. Attaching 3 to 9 violates the union-by-height rule.



(d) No. When 1 becomes the parent of 2, the height of 1 is at most 2; however, the height of 2 is 3, violating the union-by-height rule.
Note: If you answer Yes and give a sequence of operations with a find using path compression, you will get full credit.



8. (9 pts) Consider the following three min-heap implementations. Give the worst-case running time for the respective operations in $O()$.

Sol.

	delete-min	decrease-key	union
binomial heap	$O(\log n)$	$O(\log n)$	$O(\log n)$
Fibonacci heap	$O(n)$	$O(n)$	$O(1)$
skew heap	$O(n)$	$O(n)$	$O(n)$

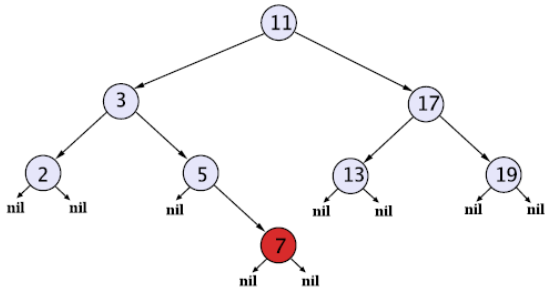
Note: For skew min-heaps, the following insertion sequence will yield a heap whose right path is of length n

$$n, n-1, n-2, ..., 2, 1, n+1, n+2, ..., 2n$$

9. (10 pts) Consider the binary search tree in Figure 3.

(a) (5 pts) Explain why this binary search tree cannot be colored to form a legal red-black tree.
Sol. We prove this by contradiction. Suppose that a valid coloring exists. In a red-black tree, all paths from a node to descendant leaves contain the same number of black nodes. The path (17, 19, NIL) can contain at most three black nodes (including the external node "NIL"). Therefore the path (17, 11, 3, 5, 7, NIL) must also contain at most three black nodes and at least three red nodes. By the red-black tree properties, the root 17 must be black and the NIL node must also be black. This means that there must be three red nodes in the path (11, 3, 5, 7), but this would mean there are two consecutive red nodes, which violates the red-black tree properties. This is a contradiction, therefore the tree cannot be colored to form a legal red-black tree.

(b) (5 pts) The binary search tree can be transformed into a red-black tree by performing a single rotation. Draw the resulting red-black tree, and label each node with "red" or "black."
Sol. There are several ways to do that. The following is one of them.



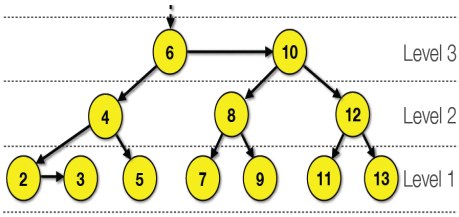


Fig. 1

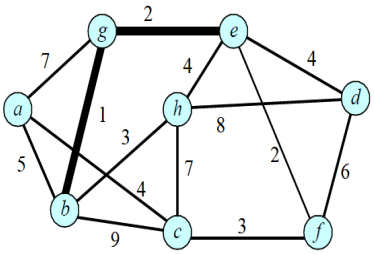
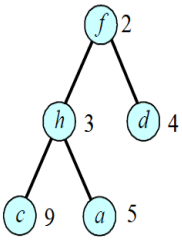


Fig. 2



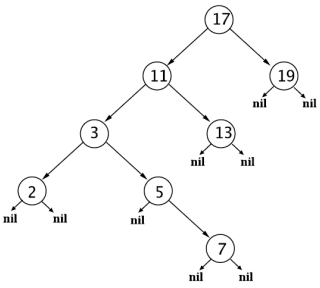


Fig. 3