York University
EECS 2011Z Winter 2015 – Problem Set 3
Instructor: James Elder

Solutions

1. Choosing a data structure

   State in one or two words the simplest ADT and implementation we have discussed that would meet each requirement.

   (a) O(1) time removal of the most recently added element
   ADT: Implementation:

   - Answer: ADT: Stack, Implementation: Array

   (b) O(1) average time addition, removal, access and modification of (key, value) pairs with unique keys
   ADT: Implementation:

   - Answer: ADT: Map, Implementation: Hash table

   (c) O(1) time insertion and removal when you are given the position
   ADT: Implementation:

   - Answer: ADT: Node List, Implementation: Doubly-linked list.

   (d) O(1) time index-based access and modification and amortized O(1) addition of elements
   ADT: Implementation:

   - Answer: ADT: Array List, Implementation: Array

   (e) O(log n) time insertion of (key, value) entries and O(log n) removal of entry with smallest key
   ADT: Implementation:

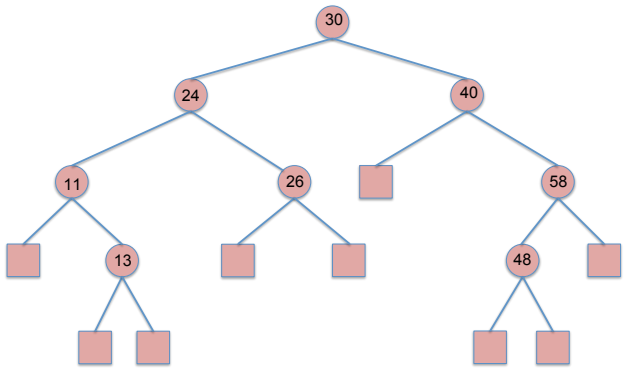
   - Answer: ADT: Priority Queue, Implementation: Heap

   (f) O(1) time removal of the least recently added element
   ADT: Implementation:

   - Answer: ADT: Queue, Implementation: (circular) array

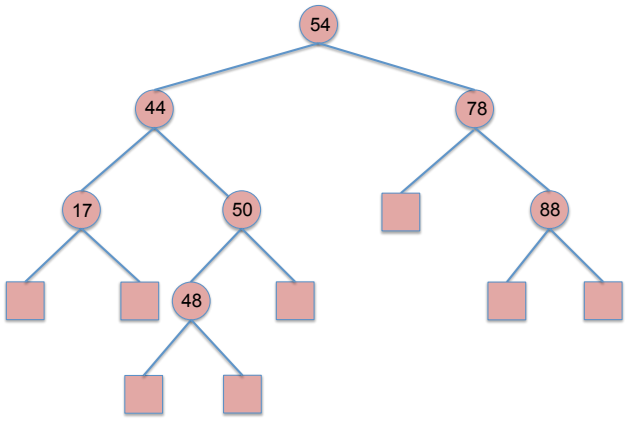
2. Binary Search Trees
   Insert, into an empty binary search tree, entries with keys 30, 40, 24, 58, 48, 26, 11, 13 (in this order). Draw the tree after each insertion.

   - Answer: The final tree should appear as follows:



3. AVL Trees
Insert, into an empty binary search tree, entries with keys 62, 44, 78, 17, 50, 88, 48, 54 (in this order). Now draw the AVL tree resulting from the removal of the entry with key 62.

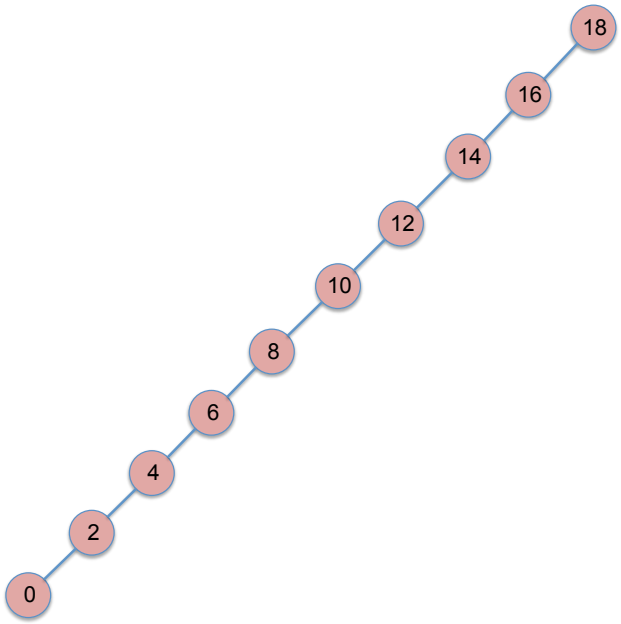
- Answer: The final tree should appear as follows:



4. Splay Trees
Perform the following sequence of operations in an initially empty splay tree and draw the tree after each set of operations.

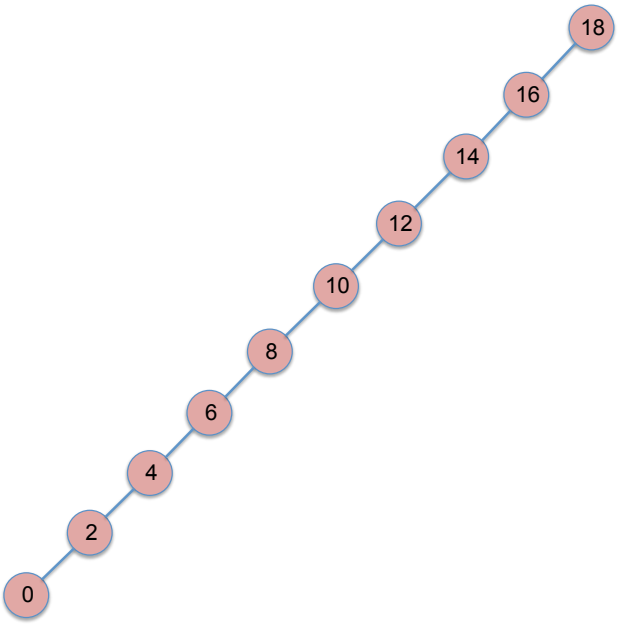
(a) Insert keys 0, 2, 4, 6, 8, 10, 12, 14, 16, 18, in this order.

- Answer:



(b) Search for keys 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, in this order.

- Answer:



(c) Delete keys 0, 2, 4, 6, 8, 10, 12, 14, 16, 18, in this order.

- Answer: The tree is empty.

5. Comparison Sorts
Of the $n!$ possible inputs to a given comparison-based sorting algorithm, what is the absolute maximum number of inputs that could be sorted with just n comparisons?

- Answer: A decision tree of height n can store 2^n external nodes. Thus a maximum of 2^n inputs can be sorted with n comparisons.

6. Comparison Sorts
Give an example input list for which merge-sort and heap-sort take $\mathcal{O}(n \log n)$ time, but for which insertion sort takes $\mathcal{O}(n)$ time. What if the list is reversed?

- Answer: Merge-sort and heap-sort both take $\mathcal{O}(n \log n)$ time on a sorted list, while insertion sort takes $\mathcal{O}(n)$ time. If the list is reversed, merge-sort and heap-sort still take $\mathcal{O}(n \log n)$ time, while insertion sort takes $\mathcal{O}\left(n^2\right)$ time.

7. Stack-Based Quicksort
Describe in pseudocode a non-recursive version of the quick-sort algorithm that explicitly uses a stack.

- Answer:

```
QuickSort(A)
stack S
push A onto S
while S ≠ ∅
        A = pop(S)
        if A.length > 1
                (L, R) = Partition(A)
                push R onto S
                push L onto S
```

8. Linear Sorts
   Given an array of n integers, each in the range $[0, n^2 - 1]$, describe a simple method for sorting the array in $\mathcal{O}(n)$ time.

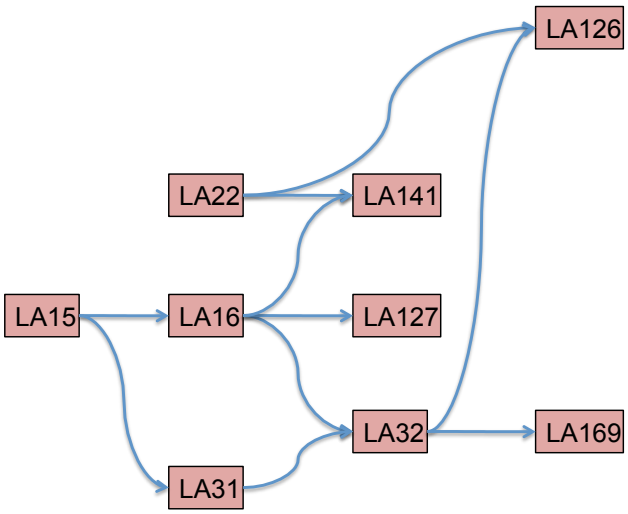
   - Answer: Notice that each integer can be coded by $\lceil \log n^2 \rceil = \lceil 2 \log n \rceil \leq 2 \lceil \log n \rceil$ bits. We choose to represent each integer by two $\lceil \log n \rceil$-bit words, each of which can assume $2^{\lceil \log n \rceil}$ values. We then use radix sort to sort the integers, which takes $O\left(2\left(n + 2^{\lceil \log n \rceil}\right)\right) \in O\left(2\left(n + 2n\right)\right) \in O\left(n\right)$ time.

9. Topological Sorts
   Suppose that you wish to take a sequence of language courses with the following prerequisites:

Course	Prerequisite
LA15	none
LA16	LA15
LA22	none
LA31	LA15
LA32	LA16, LA31
LA126	LA22, LA32
LA127	LA16
LA141	LA22, LA16
LA169	LA32

   (a) Draw a directed graph that represents these dependencies.

   - Answer:



(b) Use the topological sorting algorithm to compute a feasible sequence.

- Answer: Let's assume that vertices and edges are processed in the order indicated by the table above. Then the first call to Topological Visit will be from LA15, and will produce the list {LA15,LA31,LA16,LA141,LA127,LA32,LA169,LA126}. The second and last call to Topological Visit will be from LA22, which will simply prepend LA22 onto the list. Thus the final linear ordering will be {LA22,LA15,LA31,LA16,LA141,LA127,LA32,LA169,LA126}.

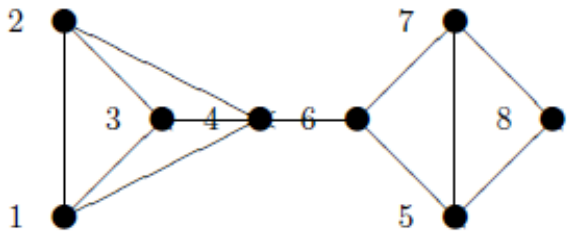
10. DFS and BFS
Let G be an undirected graph whose vertices are labelled by the integers 1 through 8, and having the following edges:

Vertex	Edges
1	2, 3, 4
2	1, 3, 4
3	1, 2, 4
4	1, 2, 3, 6
5	6, 7, 8
6	4, 5, 7
7	5, 6, 8
8	5, 7

Assume that, in a traversal of G, the adjacent vertices of a given vertex are returned in the order above.

(a) Draw G.

- Answer:



(b) Give the sequence of vertices of G visited using a DFS traversal starting at vertex 1.

- Answer: 1, 2, 3, 4, 6, 5, 7, 8.

(c) Give the sequence of vertices visited using a BFS traversal starting at vertex 1.

- Answer: The order is the same in this case: 1, 2, 3, 4, 6, 5, 7, 8