

Instructor: Alex Ryba

The 2nd midterm will cover Chapters: 1,2,3,4,5,6,7,8,10,9.

In addition to the above chapters, the final will also cover Chapter 11.

The following review problems (for the 2nd midterm and final) have appeared on old exams for this course. In addition to these review problems, you should go over the review questions that were assigned from the book to prepare for chapter review quizzes.

Problem 1 Each part of this problem gives some methods. You should answer by naming an ADT or data structure that supplies efficient implementations of the methods, specifying an appropriate implementation strategy, and writing a O -estimate for the run times of the given operations. For example, if the question read:

Methods: insert and remove using LIFO rules.

You could answer:

Stack. Linked node implementation. All operations have time $O(1)$.

Note that standard method names such as push and pop have not been used in this problem. All O -estimates should be specified in terms of n , the number of items in the structure.

- (i) Methods: insert and remove, where important items are removed first.
- (ii) Methods: insert and remove, where items that have waited the longest are removed first.
- (iii) Methods: insert, remove and find, where data items are Comparable.
- (iv) Methods: insert, remove and find, where data items are not Comparable, but include a key. (In this case you do not need to specify an implementation or run times.)

Problem 2 In each part of this problem, you are to draw a diagram showing the (legal) state of a binary tree structure that meets the indicated conditions. Each tree is to store data items consisting of positive integers.

- (a) Draw an example of a binary search tree whose preorder traversal reads: 3 1 4 7 5 6 9 8
- (b) Draw an example of a (binary) heap whose preorder traversal reads: 1 2 3 4 5 6 7 8
- (c) Draw an example that shows a correctly balanced AVL tree (with more than 2 nodes) such that insertion of the item 1 causes a rebalancing of the tree.

Problem 3 For each of the following structures list the additional requirements that a binary tree must satisfy to qualify.

- (a) A binary search tree.
- (b) A binary heap.
- (c) An AVL Tree.
- (d). Implement the following method:

```
int height(BNode n)
```

The method should return the height of a BNode n in a binary tree.

Problem 4 Write a Java interface for each of the following ADTs. (Just write an interface that specifies important methods — **do not** implement the methods.)

- (i) The ADT Queue.
- (ii) The ADT Iterator.
- (iii) The ADT Comparable.
- (iv) The ADT PriorityQueue.
- (v) The ADT Map.
- (vi) The ADT Stack.

Problem 5 Give useful O -estimates of the run times of the following methods:

- (a) The method *insert* for a binary heap that has size n .
- (b) An efficient method to calculate the power x^n (consider the run time as a function of n , the time should be considered as being proportional to the total number of additions, subtractions, multiplications, and divisions performed).
- (c) An efficient method to sort an array of n numbers into order.

For (d) and (e), consider the following recursive function, in which A represents an integer constant:

```
int f(int n) {
    if (n <= 0) return 1;
    int ans = 2;
    for (int i = 1; i <= n; i++)
        for (int j = i; j <= n; j++)
            ans += i / j;
    for (int k = 1; k < A; k++)
        ans -= f(n/2 - k);
    return ans;
}
```

- (d) In the case where $A = 3$ estimate the run time of $f(n)$.
- (e) In the case where $A = 4$ estimate the run time of $f(n)$.

Problem 6 In each part of this problem, you are to draw one or more diagrams showing legal state(s) of a binary tree structure that meets the indicated conditions. Each tree is to store data items consisting of positive integers.

- (a) Draw an example of a binary search tree whose postorder traversal reads: 2 1 6 9 5 4 3
- (b) Draw two different examples of a binary search tree whose inorder traversal reads: 1 2 3 4 5 6 7 8 9
- (c) Draw an example that shows a correctly balanced AVL tree with that property that removal of the item 1 causes more than one rebalancing operation of the tree. Also draw the state of the tree after these rebalancing operations have been completed.

Problem 7 A midOrder traversal of a GeneralTree processes a node that has n children after processing the subtrees headed by the first $n/2$ children and before processing the subtrees headed by the remaining children.

Assume that GeneralTrees are implemented as in class, based on the following skeleton Java code for GeneralTrees and GNodes.

```
class GNode extends Node {
    // The inherited instance variables from class Node are: parent and data
    Vector children; // can be empty, but is never null
    // standard methods and constructors omitted, except
    public Iterator children() { return children.iterator(); }
}

class GeneralTree extends Tree {
    // The inherited instance variable from class Tree is: Node root
    // standard methods and constructors omitted
}
```

Suppose that the class GeneralTree includes the method:

```
public Vector midOrder() {
    Vector answer = new Vector();
    if (root != null) midOrder((GNode) root, answer);
    return answer;
}
```

Write a Java implementation for the auxiliary recursive method called `midOrder`.

Answer:

Problem 8 Write a complete Java program that reads a file of text (specified as the first command line argument) and prints out the commonest word. The commonest word is the word that appears most often in the text. In case several different words tie as the commonest word, print all of them. (I suggest that you write a program that makes use of the Java interface `Map`.)

As an example, if the text reads:

```
the quick brown fox jumps over and over the lazy dog
```

The output should read:

```
Commonest word(s):  the over
```

Answer:

Problem 9 For each of the following ADTs and data structures, specify the most important operations (methods). For each method, write your answer as a title line such as:

```
public Object pop()
```

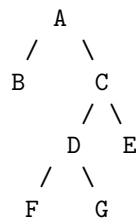
(i) Deque (4 operations)

(ii) Priority Queue (2 operations)

(iii) Map (3 operations)

(iv) Binary Search Tree (3 operations)

Write the Euler tour traversal of the following binary tree



Problem 10 (a) The data items 1, 2, 3, 5, 4, 7, 6 are to be inserted in this order into an AVL tree. Draw 7 tree diagrams that show the state of the structure after each of the 7 insertions.

(b) Suppose that a heap is implemented in the usual way (which we covered in class) to store its entries in an array. The root of the heap is to be stored at index 0. Later elements are stored so that a level order traversal is given by the array elements with indices 0, 1, 2, ...

Write the indices of the children of a node with index n .

Write the index of the parent of a node with index n .

Write a boolean expression that uses the index n of a node and $size$ (the number of items in the heap). The expression should be true exactly when n is the index of a leaf node.

Problem 11 The following is a skeleton of the Binary Search Tree implementation that we considered in class.

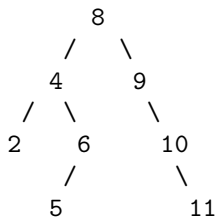
```

class BST extends BinaryTree {
    public BNode findNode(Comparable x) {
        if (root() == null) return null; return findNode((BNode) root(), x); }
    public BNode findNode(BNode n, Comparable x) {
        Comparable y = (Comparable) n.getData();
        if (x.compareTo(y) < 0 && n.getLeft() != null) return findNode(n.getLeft(),x);
        if (x.compareTo(y) > 0 && n.getRight() != null) return findNode(n.getRight(),x);
        return n;
    }
    public void insert(Comparable x) {
        BNode n = findNode(x); if (n == null) addRoot(x);
        else if (((Comparable) n.getData()).compareTo(x) > 0) addLeft(n, x);
        else if (((Comparable) n.getData()).compareTo(x) < 0) addRight(n, x);
    }
    /* other methods, not relevant to this problem, are omitted */ }

```

The implementation does not add duplicate data into the tree. The goal of this question is to modify the insert method to allow duplicate entries in the tree. If the parameter x is found in the tree, then a second or later copy of x is to be inserted as a leaf in the left subtree of the node that contains the earlier copy of x .

(a) On the following diagram, indicate the only location where the insertion of a second copy of the entry 8 could be made.



(b) Write a modified version of the **insert** method which will allow duplicates to be inserted. (Does it carry out the insertion at the indicated position of the diagram in (a)?)

(Extra credit) If a binary search tree can contain duplicated data items, it can be used to implement a priority queue. The binary search tree already has an insert method. Write an additional **BST** method called **removeMin** that would complete an implementation of the interface **PriorityQueue**. Write your extra credit implementation on the facing page.

Problem 12 Consider the following diagram showing the state of a binary tree.

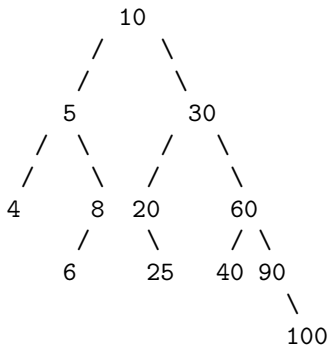


Diagram T.

- (a) Write down the inorder traversal of the tree.
- (b) Write down the preorder traversal of the tree.
- (c) Write down the Euler tour traversal of the tree.
- (d) Assume that Diagram T represents the state of an AVL Tree. What data item could be inserted to unbalance the tree? Show the steps involved in rebalancing the tree after the insertion of this item.
- (e) Again assume that Diagram T represents the state of an AVL Tree. Show the steps involved in rebalancing the tree after the removal of the Node storing 4.

Problem 13 A presidential campaign has collected a large data file of phone pledges of money. In the file the record of each phone conversation appears on a line that begins with a # sign and then has a 10 digit string representing a phone number (and no other text). The record includes any useful facts from the conversation and most importantly any dollar amounts pledged. These dollar amount appear as strings following a \$ sign (also on their own lines). The characters # and \$ are only ever used to start a phone number or dollar amount.

A typical segment of the file might read as follows:

```
#2120011234
Will give more if promised a casino license.
$2.37
#2120011235
Go away
#2120011236
Worked for us before.
$500
Wants a presidential pardon.
$20000
Volunteers to work for the campaign if bailed.
$500
#2120011237
```

Consider the following skeleton of a program to process the data.

```
import java.util.*;
import java.io.*;

class Donations {
    public static void main(String args[]) {
        try {
            Scanner inData = new Scanner(new File(args[0]));
            Map cash = new HashMap();
            findMoney(cash, inData);
            showMoney(cash);
        }
        } catch (Exception e) {}

    public boolean isPhone(String s) // returns true if s represents a Phone number
    public boolean isMoney(String s) // returns true if s represents a Dollar amount
    // code omitted, but the methods should be used to help your solutions

    // other methods appear here
}
```

(a) Write the method *findMoney*. The goal of the method is to store any dollar amounts found on the Scanner as values in the map. No other information from the Scanner is to be stored in the map. Each value is to use a phone number as its key. The first entry stored in the example above should be stored under the key "#2120011234" as the entry "\$2.37". (Make sure that new entries for a key are appended after prior entries.)

Write your solution on the facing page.

(b) Write the method *showMoney*. The goal of this method is to make a short list of pledged contributions. An example of output follows:

```
#2120011234 $2.37
#2120011236 $500 $20000 $500 $1000 $250
```

Write your solution on the facing page.

Extra credit: Write a method to list all numbers for which the same amount appears twice as a pledge. In the above example, #2120011236 should be listed because \$500 is pledged twice from there. (Hint: More maps.)

Problem 14 For each of the following structures list the additional requirements that a binary tree must satisfy to qualify.

- (a) A binary search tree.
- (b) A binary heap.
- (c) An AVL Tree.
- (d). Implement the following method:

```
boolean isBST(BinaryTree t)
```

The method should examine a Binary Tree *t* and determine whether it is a binary search tree.

Hint: Is there a standard binary tree method that can be used to make the definition of a binary search tree?

Problem 15 Suppose a heap based Priority Queue is programmed as in class. (So that the minimum element is found at the root node, and is stored at array location 0.) The following is a skeleton of the implementation:

```
class PriorityQueue {
    private Comparable data[];
    private int size;
    // The following method are present, but their code is omitted
    public PriorityQueue() ...
    public Comparable removeMin() ...
    public void insert(Comparable x) ...
    public void bubbleUp(int n) ...
    public void bubbleDown(int n) ...
}
```

Write an additional (non-standard) method:

```
Comparable returnMax()
```

The method is to find and return the value of the largest item in the PriorityQueue without making any alteration to the queue itself.

Problem 16 Write a program that reads a file of words and prints a list of all the different words together with their frequency. For example, a segment of output might look like:

```
dog 2
the 425
and 200
of 23
is 120
java 5
```

I suggest that you use a Map that uses words as keys and keeps a count for the word as a value. Your main program might use the following skeleton to open an input file and read words:

```
public static void main(String args[]) {
    try{
        Scanner s = new Scanner(new File(args[0]));
        // set up the Map and other variables here
        while (s.hasNext()) {
            String word = s.next();
            // store the word
        }
        // retrieve and output data
    } catch (Exception e) {}
}
```

Problem 17 For each of the following structures list the additional requirements that a binary tree must satisfy to qualify.

(a) A binary search tree.

(b) A binary heap.

(c) An AVL Tree.

(d). Implement the following method:

```
int size(Node n)
```

The method should return the number of descendants of a Node n in a binary tree. (The node itself should be counted among the descendants unless it is null.)

Problem 18 (a) Give useful O -estimates for run times of each of the following (define any parameters such as n or h that appear in your estimates):

- Insert a Comparable object into an AVL Tree.
- Enqueue, for a Queue, circular array implementation:
- Find, for a List, linked node implementation:
- Insert a Comparable object into a Priority Queue, heap implementation.
- Find a key from a chained Hash Table with N items and load factor L .
- Merge sort an array with n items of data.

Problem 19 The data items 6, 7, 8, 3, 5, 1, 9 are to be inserted in this order and then 1 is to be removed using each of the following data structures. Draw diagrams (8 per structure) to show the state of the structure after each of the 7 insertions and after the final removal.

- (a) The structure is a Binary Search Tree
- (b) The structure is a heap.
- (c) The structure is an AVL Tree
- (d) The structure is an initially empty hash table that resolves collisions by linear probing. The table has a capacity of 7 and we use a hash function that takes the remainder when an integer is divided by 5. (For example, 196884 hashes to 4.)

Problem 20 For each of the following ADTs and data structures, specify the most important operations (methods). For each method give the name, parameters and return type. For each method give a O estimate (in terms of the size n) for the run times of efficient implementations of the operations.

- (i) AVL Tree (3 operations)
- (ii) Deque (4 operations)
- (iii) Map (3 operations)
- (iv) Stack (2 operations)
- (v) Iterator (2 operations)
- (vi) Comparable (1 operation)

Problem 21 For each part, specify the additional properties of a binary tree that are required to say that it has the given structure.

- (i) Binary Search Tree
- (ii) Heap
- (iii) AVL Tree

Problem 22 The following skeleton of a class Queue is missing the methods enqueue and dequeue. Give Java implementations for the methods. The queue is to be implemented with singly linked nodes. You do not need to implement the Node class, and you may assume that it has standard methods.

```
class Queue {
    protected Node front, rear;
    private int size;
    public Queue() {
        front = rear = null;
        size = 0;
    }
    public int size() {
        return size;
    }
    public boolean empty() {
        return size == 0;
    }
    public Object dequeue() // give an implementation for this
    public void enqueue(Object x) // give an implementation for this
}
```

Problem 23 The following skeleton of a class implements merge sort. The merge method is missing. Give a Java implementation for the merge method.

```

class MergeSort {
    public Comparable a[];
    public int capacity;

    // a missing constructor sets the array a[] and its capacity
    // missing methods copy data to be sorted into the array a[]

    public void mergeSort(int low, int high){
        if (low >= high) return;
        int mid = (low + high + 1) / 2;
        mergeSort(low, mid - 1); mergeSort(mid, high);
        merge(low, mid, high);
    }

    public void merge(int low, int mid, int high) // add an implementation below
}

```

Problem 24 The data items 5, 8, 7, 4, 6, 2, 9 are to be inserted in this order and then 2 is to be removed using each of the following data structures. Draw diagrams (8 per structure) to show the state of the structure after each of the 7 insertions and after the final removal.

- The structure is a Binary Search Tree
- The structure is a heap.
- The structure is an AVL Tree
- The structure is an initially empty chained hash table of capacity 5 that uses a hash function that takes the remainder when an integer is divided by 5. (For example, 196884 hashes to 4.)

Problem 25 For each of the following ADTs and data structures, specify the most important operations (methods) and give a O estimate (in terms of the size n) for the run times of efficient implementations of the operations.

- AVL Tree (3 operations)
- Priority Queue (2 operations)
- Map (3 operations)

Draw the expression tree for the following algebraic expression:

$$2 / (2 * z + 2 / (2 * y + 3 * x))$$

Problem 26 The following skeleton of a Binary Search Tree class describes two methods (find and remove). Give Java implementations for the two methods (include code for any auxiliary helper methods that you make use of).

```

public class Node { // stores the left, right children and the parent
    public Node left, right, parent;
    public Comparable element;
}

public class BinarySearchTree {
    // only 2 instance variables as follows
    private Node root;
    private int size;

    /* Methods include ---
    public boolean find (Comparable x) // returns true if x is stored in the tree
    public void remove(Comparable x) // removes x from the tree if x is already present
    */
}

```

Problem 27 The data items 2, 7, 1, 8, 28, 18, 2845 are to be inserted in this order into each of the following data structures. Draw diagrams (7 per structure) to show the state of the structure after each insertion.

(a) The structure is a Binary Search Tree

(b) The structure is a heap.

(c) The structure is an initially empty chained hash table of size 10 that uses a hash function that takes the last digit of an integer. (For example, 196884 hashes to 4.)

(d) The structure is an initially empty (unchained) hash table of size 10 that uses a hash function that takes the last digit of an integer and resolves collisions by linear probing.

Problem 28 The following skeleton of a Binary Search Tree class describes two methods (find and insert). Give Java implementations for the two methods (include code for any auxiliary helper methods that you make use of).

```
public class Node { // stores the left, right children and the parent
    public Node left, right, parent;
    public Comparable element;
}

public class BinarySearchTree {
// only 2 instance variables as follows
    private Node root;
    private int size;

/* Methods include ---
    public boolean find (Comparable x) // returns true if x is stored in the tree
    public void insert(Comparable x) // inserts x into the tree unless x is already present
*/
}
```

Problem 29 (a) Give useful O -estimates for run times of each of the following (define any parameters such as n or h that appear in your estimates):

- Insert a Comparable object into a balanced Binary Search Tree.

- Enqueue, for a Queue, circular array implementation:

- Find, for a List, linked node implementation:

- Insert a Comparable object into a Priority Queue, heap implementation.

- Find a key from a chained Hash Table with N items and load factor L .

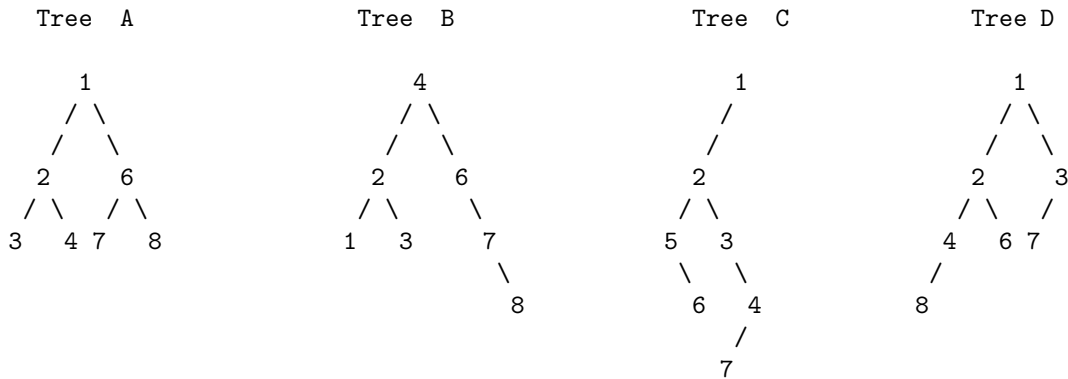
(b) What two properties of a binary tree make it a heap?

(c) Give either a pseudocode outline (or for extra credit, a Java method) for an algorithm:

```
boolean hasHeapShape(Node r)
```

That returns true, if the subtree rooted at r satisfies the heap shape requirement.

Problem 30 Consider the following representations of Trees.

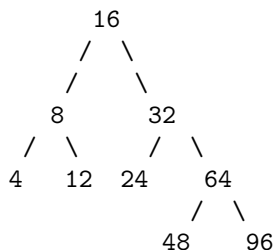


- Which trees satisfy binary search ordering?
- Which trees are heaps?
- Which Nodes have depth 2?
- Which Nodes have height 2?
- Write the PreOrder traversal of Tree A:
- Write the PostOrder traversal of Tree B:
- Write the Inorder traversal of Tree C:
- Write the level order traversal of Tree D:
- What would become the new root of the Tree that represents a heap, after the operation Remove Min?
- What would be the new height of the Tree that represents a binary search tree if 10 was inserted?
- Is there a heap storing seven distinct elements such that a preorder traversal yields the elements in sorted order?
- Is there a heap storing seven distinct elements such that a preorder traversal does not yield the elements in sorted order?

Problem 31 Give a complete Java implementation of a Stack ADT.

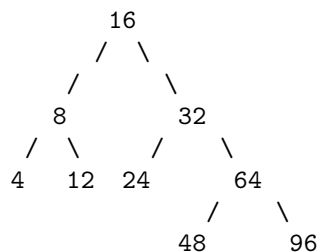
Problem 32 (a) Suppose that the data in a *Node* n is to be removed from a binary search tree. Give a pseudocode algorithm to remove the data and reshape the tree if necessary.

(b) Suppose that the data item 16 is to be removed from the following Binary Search Tree. Give a sequence of diagrams to show how your algorithm of (a) would remove the data item and reshape the tree.



Problem 33 (a) Suppose that a *Node* n of an AVL Tree is unbalanced and that $height(n.right) == height(n.left) + 2$. Give a pseudocode algorithm to rebalance the node.

(b) Suppose that the data item 40 is added to the following AVL Tree. Give a sequence of diagrams to show how the data is added and how balance is then restored to the tree. Indicate which nodes need to be rebalanced with your algorithm of (a).



Problem 34 (a) Describe the instance variables that are used to store and implement a chained hash table. (Give Java declarations for the instance variables.)

(b) Using the instance variables that you gave in (a), implement the methods *insert(Key k, Object x)* and *Object find(Key k)*.

Problem 35 The data items 2, 7, 1, 8, 28, 18, 2845 are to be inserted in this order into each of the following data structures. Draw diagrams (7 per structure) to show the state of the structure after each insertion.

(a) The structure is a Binary Search Tree

(b) The structure is a heap.

(c) The structure is an initially empty chained hash table of size 10 that uses a hash function that takes the last digit of an integer. (For example, 196884 hashes to 4.)

(d) The structure is an initially empty (unchained) hash table of size 10 that uses a hash function that takes the last digit of an integer and resolves collisions by linear probing.

Problem 36 A failed internet business needs to wind up their operations. Please advise them on the best data structures to use in implementing programs for the following tasks. Give brief explanations of your advice in each case.

(a) Maintain a searchable catalogue of company property for sale. The user chooses from a general list of categories, and successively more precise submenus are made available. For example, the a menu might include company cars, computers, and office furniture. If a user selects computers, the next menu could give a choice between PCs or Macs.

(b) Maintain an inventory of company property. Items will be looked up by serial number and removed from the inventory as they are sold off.

(c) Keep a record of creditors to be paid off. Important creditors are to be paid off first, and payments are to be made as soon as funds are available.

(d) Set up a record of employees to lay off. Enter details of long term employees first, but lay off the most recent employees in the record whenever cuts are needed.

(e) Keep a record of loyal customers, organized by 9 digit zip code. A customer should be able to call in and say: "My zip code is 11367-1234. Give me the names of customers who live nearby, who could help me to lobby Senator Clinton to bail you out."

Problem 37 The following data items are to be added to an initially empty structure in the order that they are listed.

5 1 9 3 4 2

Draw a sequence of diagrams (one diagram per item) to show how the structure grows when:

(a) The structure is a heap.

(b) The structure is a binary search tree.

(c) Give a pseudocode outline of an efficient method to test whether a Binary Tree has the heap ordering property.

(d) Give a pseudocode outline of an efficient method to test whether a Binary Tree has the binary search tree ordering property.

Problem 38 Write a Java implementation of a Binary Search Tree method, *find* that searches the tree for a Comparable object (supplied as a parameter).

Problem 39 Write a Java implementation for a method to be called *locateData*. The method *locateData* has a single parameter representing an array of Integer values. (The array can be assumed to have about a million entries.) The method *locateData* runs a loop in which a user asks questions about the stored data. The user enters a question by typing an integer. The program responds either by stating that the integer is present in its data, or by returning the closest integer that is present in its data. For example, a pair of questions might appear as:

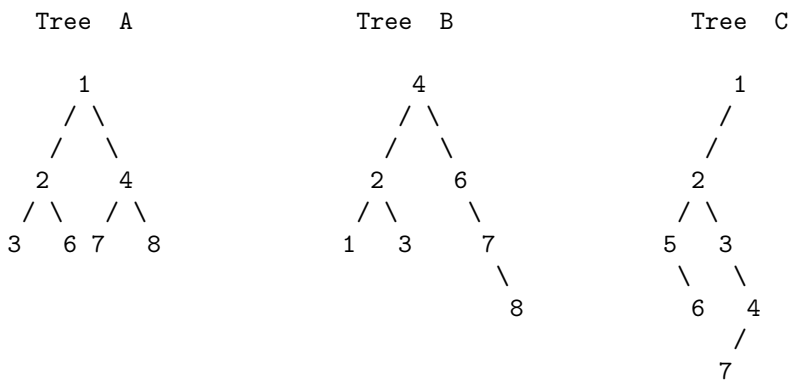
```
What key number do you want to search for? 196884
That is present.
What key number do you want to search for? 21296000
The closest key is 21296876
```

Your program should begin by copying the data in the array to a more appropriate data structure of your choice. The program should then run a loop that answers user questions. Your program should operate reasonably efficiently. You may use any of the data structures that we have discussed and implemented. You do not need to implement the data structures that you use.

Problem 40 Give O -estimates for the average run time of each of the indicated methods:

- Push, for a Stack of N items, array implementation:
- Dequeue, for a Queue of N items, circular array implementation:
- Height, for a BinaryTree of N items, linked node implementation:
- RemoveMin, for a PriorityQueue of N items, heap implementation:
- InsertionSort, of an array of N items:
- HeapSort, of an array of N items:
- MergeSort, of an array of N items:
- Hash an object for a chained HashTable with N items and load factor L :
- Remove an object from a chained HashTable with N items and load factor L :
- Insert for a balanced Binary Search Tree with N items:
- Find for a balanced Binary Search Tree with N items:
- Insert all of the keys 1, 2, 3, ... N (in this order) to a Stack:
- Insert all of the keys 1, 2, 3, ... N (in this order) to a Priority Queue, heap implementation:
- Insert all of the keys 1, 2, 3, ... N (in this order) to a Hash Table with N buckets:
- Insert all of the keys 1, 2, 3, ... N (in this order) to a Hash Table with 5 buckets:
- Insert all of the keys 1, 2, 3, ... N (in this order) to a Binary Search Tree:

Problem 41 Consider the following representations of Trees.



- Which trees satisfy binary search ordering?
- Which trees satisfy heap ordering?
- Which Nodes have depth 2?

- Which Nodes have height 2?
- Write the PreOrder traversal of Tree A:
- Write the PostOrder traversal of Tree B:
- Write the Inorder traversal of Tree C:
- What would become the new root of the Tree that represents a heap, after the operation Remove Min?
- What would become the new root of the Tree that represents a binary search tree if its root was removed according to the algorithm that we examined in class?
- Is there a heap storing seven distinct elements such that a preorder traversal yields the elements in sorted order?
- Is there a binary search tree storing seven distinct elements such that a preorder traversal yields the elements in sorted order?

Problem 42

- Give three advantages of chaining over open addressing (for hash tables).
- Give one advantage of using postfix rather than infix notation for mathematical expressions.
- Give one advantage of using infix rather than postfix notation for mathematical expressions.
- In a circular array with size N , which element follows the element numbered i ?
- Estimate T_1/T_2 where T_1 is the run time of heap sort and T_2 is the run time of selection sort, and both sorting methods are applied to 1000 data items. Give your answer as a real number.
- Estimate T_1/T_2 where T_1 is the run time of heap sort and T_2 is the run time of selection sort, and both sorting methods are applied to 10^9 data items. Give your answer as a real number.

Problem 43

A new company, QCTickets, provides software for applications to ticketing of sports events. Advise QCTickets on the best data structures to employ in the following tasks. Give brief explanations of your advice in each case.

- (a) Ticket orders are to be taken by phone. Incoming calls are put on hold until an operator is available and are then answered. What data structure should store the incoming calls?
- (b) After a ticket is ordered by phone, a request is sent to a mailing department. The mailing department prints and mails some tickets once every weekday. They start with the most expensive waiting order. What data structure should store the requests to the mailing department?
- (c) When a person uses a ticket, their ticket number must be compared quickly against the numbers of all tickets that have already been used. (To prevent people from using photocopied tickets.) How should the numbers of tickets that have been used be stored?
- (d) In order to arrange car pooling opportunities, interested people are to be notified of other customers who live in nearby zip codes. How should a list of customers and their addresses be stored?

Problem 44 Give useful O -estimates for run times of each of the following:

- Merge sort of an array of N Comparable objects.
- Insert a Comparable object into a balanced Binary Search Tree with size N .
- Enqueue, for a Queue of N items, circular array implementation:
- Find, for a List of N items, linked node implementation:
- Insert a Comparable object into a Priority Queue with size N , heap implementation.
- FindMin, for a balanced Binary Search Tree with size N .

- Remove a Hashable object from a chained Hash Table with N items and load factor L .

- Suppose that the method call `insertionSort(a, n)` carries out an insertion sort on the first n elements of an array `a` that contains `int` values. Give a useful O -estimate for the run time of the following block of code:

```
for (int i = 0; i < N; i++) {
    a[i] = (int) (Math.random() * N);
    insertionSort(a, i);
}
```

- Suppose that the method call `mergeSort(a, n)` carries out a merge sort on the first n elements of an array `a` that contains `int` values. Give a useful O -estimate for the run time of the following block of code:

```
for (int i = 0; i < N; i++) {
    a[i] = (int) (Math.random() * N);
    mergeSort(a, i);
}
```

- Give a useful O -estimate for the run time of `fun(N)`, where:

```
public static int fun(int n) {
    if (n < 10) return 1;
    return 1 + fun(n / 2);
}
```

Problem 45 The data items 3, 1, 4, 15, 9, 2, 6 are to be inserted in this order into each of the following data structures. Draw diagrams (7 per structure) to show the state of the structure after each insertion.

- The structure is an initially empty Stack
- The structure is a Queue, with a circular array implementation. Where the data array size is 11, and initially, a single value 1, is stored in the middle location of the data array.
- The structure is a heap that initially contains the values 18, 28, 27 in this order.
- The structure is an initially empty probed hash table of size 10 that uses a hash function that squares an integer value. (For example, 4 hashes to 16.)

Problem 46 Give an implementation of heap sort. (Assume that any required data structures are available.) Give a O -estimate for the run time, and show how this estimate is obtained.

Problem 47 The following skeleton of a `BinaryTree` class describes four methods (`depth`, `findMin`, `successor`, `isBST`). Give efficient Java implementations for the four methods.

Considerable partial credit will be given for good pseudo-code implementations. (Hint: use the `successor` method to write `isBST`.)

```
public class Node { // stores the left, right children and the parent
    public Node left, right, parent;
    public Comparable element;
}
```

```
public class BinaryTree {
```

```
public Node root;

/* Methods include ---
public int depth(Node n) // returns the depth of Node n
public Node findMin(Node n) // returns the Node in the subtree of Node n
// that has the smallest element
public Node successor(Node n) // returns the Node that would follow Node n
// in an inOrder traversal of the Tree
public boolean isBST() // returns true if the data stored in the Tree
// satisfies the Binary Search Tree property.
*/
}
```