**NAME:**

**COMP231 – Data Structures and Problem Solving**

**Spring 2019**

**Final Exam**

**May 9, 2018 9:00am**

1. The sumInc5s method is to answer the following question. Given an array of integers, is it possible to choose some of the elements such that they sum to a given target value, with the following additional constraint: When a 5 is encountered it must be chosen. For example:

[4, 5, 3, 2, 7] with target 15 → true  
[4, 5, 3, 5, 7] with target 15 → false

[2] a. Complete the sumInc5s method shown below assuming that the sumInc5sHelper method defined in part b works.

**public** **static** **boolean** sumInc5s(**int**[] vals, **int** targetSum) {

sumInc5sHelper( vals, targetSum, 0, 0);

}

[10] b. The sumInc5sHelper method below solves a transformed version of the original problem. Complete the implementation of sumInc5sHelper so that the sumInc5s method from part a will answer the original question.

**public** **static** **boolean** sumInc5sHelper( **int**[] vals, **int** targetSum,

**int** curSum, **int** start) {

if( curSum == targetSum){

return true;

}

else if( start == vals.length){

return false;

else{

boolean take = sumInc5sHelper(vals, targetSum, curSum+ vals[start], start+1;

Boolean leave = false;

If(vals[start]!=5){

leave = sumInc5sHelper( vals, targetSum, curSum, start+1);

}

return take||leave;

}

[5] 2. Consider the following basic operation count for the worst case of an algorithm:

T(n) = 2n2 + 19n + 7

Use the formal definition of Ο to show that the algorithm is in Ο(n3).

lim n->infinity (2n2+19n+7)/(n3) = 0. Since the limit is not infinity, 2n2+19n+7 is in n3.

3. Consider the following recurrence relation:

T(1) = 3

T(n) =3 T(n-1) + n

[5] a. Show at least 2 steps in the expansion of T(n).

T(n) = 3T(n-1) + n

= 3( 3T(n-2)+n-1) +n=9T(n-2)+4n-3

=9( 3T(n-3) + n-2) + 4n-3=27T(n-3) +13n-21

=3k T(n-k) + sum{i=1 to k) 3i – sum 3i-1

[2] b. Give a theta value of T(n).

The final expression from above is in theta(3k).

4. The BTNode class that we used in our binary tree and BST implementations is shown below:

**static** **protected** **class** BTNode<K, V> {

**public** K key;

**public** V value;

**public** BTNode<K, V> left;

**public** BTNode<K, V> right;

**public** BTNode<K, V> parent;

**public** BTNode(K key, V value) {

**this**.key = key;

**this**.value = value;

left = **null**;

right = **null**;

parent = **null**;

}

**public** **boolean** isLeaf() {

**return** left == **null** && right == **null**;

}

}

Complete the methods defined in each of the following questions. Assume that each method is called with the root node of a binary search tree (BST) with at least 1 node in both its left and right sub-trees (i.e. root.left!=null and root.right!=null). For full credit, solutions must make effective use of the tree structure and the BST properties as appropriate.

[10] a. getDepth that computes the depth of the tree, where depth is the maximum distance from the root to a leaf.

**public** **static** **int** getDepth(BTNode<K,V> root) {

if(root == null){

return -1;

}

else{

return 1+ Math.max(getDepth(root.right, root.left));

}

[10] b. printReverseOrder that prints the keys in the BST from largest to smallest.

**public** **static** **void** printKeysReverseOrder(BTNode<K,V> root) {

if(root == null){

return;

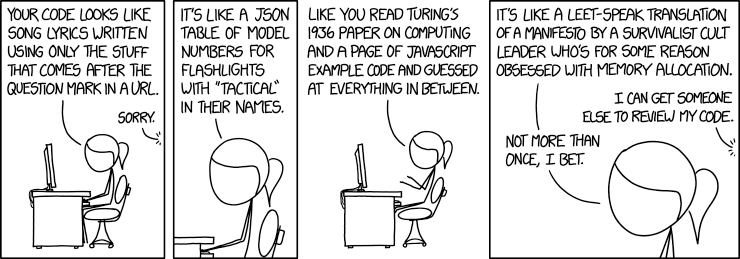
} else{

printKeysReverseOrder(root.right);

System.out.println(root.key);

printKeysReverseOrder(root.left);

}



4. Amelia’s bookshelf has n very meticulously sorted books, except for one problem. The last time she took out a book, she put the book back at the front of the shelf. Now, she wants to apply a sorting algorithm to sort the books again.

1. [5] What is the running time of selection sort on this input? Briefly explain why.

n2. Selection sort is always n2.

1. [5] What is the running time of insertion sort on this input? Briefly explain why.

n. Each insertion will have at most 2 comparisons: one to move over the misplaced book, and one to stop.

1. [5] What is the running time of merge sort on this input? Briefly explain why.

n lg n. Merge sort is always n lg n.

5. For the undirected graph shown, list the vertices in the order that they would be processed by the searches indicated below. Assume that each search starts at vertex 1 and that neighbors are considered in increasing numeric order.



[4] a. Depth-first search

1, 7, 18, 12, 27, 3, 14, 91, 19, 9, 55

[4] b. Breadth-first search

1, 7, 14, 19, 27, 18, 91, 9, 55, 3, 12

6. Briefly but precisely describe how you could perform the following operations on an undirected graph using the indicated representation(s). For each, also give the most appropriate asymptotic bound (Ω and O, or Θ) for the running time required for the operation. It is not necessary to give code for these questions.

[4] a. hasEdge(v1,v2) that returns true if edge (v1,v2) exists in the graph and false if it does not, when the graph is represented using an adjacency matrix.

Check the matrix. theta( 1).

[4] b. hasEdge(v1,v2) when the graph is represented using an adjacency list.

Check the appropriate list. O( |V | )

[4] c. isIsolated(v) that returns true if vertex v is *isolated* (a vertex is isolated if it has no incoming and no outgoing edges) when the graph is represented using an adjacency matrix.

Check the v column. The vertex is isolated if the column is all 0s. O( |V|)

[4] d. isIsolated(v) when the graph is represented using an adjacency list.

Check all edges in all lists. The vertex is isolated if none of them reach v. O( |E|)

7. [6]Draw the graph represented by this matrix

(0 | 1 | 1 | 0
1 | 0 | 0 | 1
1 | 0 | 0 | 0
1 | 1 | 0 | 0)

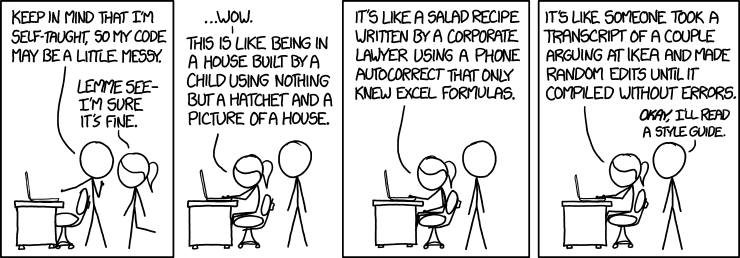
8. Consider a hash map with hash function h(k)=k % tableSize. Assume that tableSize=10 and the hash map will not be resized. Show the contents of the hash table if the following keys are inserted in order {57, 22, 87, 88, 257} under the following conditions:

1. [10]Open Hashing / closed addressing.

(x,x,(22),x,x,x,x,(57,87,257), (88),x,x)

1. [10] Closed hashing / open addressing with the quadratic probe function p(k,i)=(i2+i)/2.

(257,x,22,x,x,x,x,57,87,88)



[15] 9. Circle T or F to indicate if each of the following statements is true or false:

T / **F** In a max heap with an array backing store, two identical keys will never appear in consecutive array indices.

T / **F** In a postorder traversal of a max heap, the values will be visited in ascending order.

**T** / F In a preorder traversal of a max heap, the first item visited will always be the largest one.

**T** / F A heap containing n elements can be constructed in O(n) time.

T / **F** A BST guarantees that any key in the tree can be found in O(lg n) time.

**T** / F With open hashing the load factor can be greater than 1, but in closed hashing the load factor must never be allowed to be greater than 1.

T / **F** Every node in a B+Tree is at least ½ full.

T / **F** Insertion sort runs in O(n2) time if the list is already in sorted order.

T / **F** The best case input for the sorting problem is if the list is already sorted.

**T** / F Merge sort will run in O(n lg n) time on any input.

For the following questions, assume that A and B are two algorithms for solving the same problem.

**T** / F If A is in Ω(n) and B is in Θ(lg n) then there must be an instance of the problem for which B will run faster than A.

**T** / F If A is in Ο(n) and B is in Ω(n lg n) then there must be an instance of the problem for which A will run faster than B.

T **/ F** If A is in Θ(n) and B is in Θ(nlg n) then A will always run faster than B.

**T** / F If A is in Ω(n lg n) and B is in O(n) then there must be an instance of the problem for which B will run faster than A.

T / **F** If A is O(n2) and B is O(n), there must be an instance of the problem where B runs faster than A.

**public** **interface** CS232Graph<V, E> {

**public** **int** numVertices();

**public** **int** numEdges();

**public** V getVertexObject(**int** v);

**public** **void** setVertexObject(**int** v, V value);

**public** **int** getVertexMark(**int** v);

**public** **void** setVertexMark(**int** v, **int** mark);

**public** **void** addEdge(**int** v1, **int** v2, E value);

**public** E getEdge(**int** v1, **int** v2);

**public** E removeEdge(**int** v1, **int** v2);

**public** ArrayList<Integer> getNeighbors(**int** v);

}

10. [15] Above is the interface for graph. Write a method that takes as an argument an integer n and returns a graph with n vertexs. In this new graph, there is an edge from a to b if and only if b is divisible by a. The weight of that edge should be b/a. (Bonus: make this function run in O(|E|) time and say what the running time is in terms of |V|).

**public** **static** CS232Graph makeGraph(**int** n){

CS232Graph g = **new** CS232DirectedAdjacencyListGraph(n);

for(int i = 0; i < n; i++){

g.addEdge(i,0,0);

}

for(int i = 1; i < n; ++i){

for(int j = i; j < n; j+=i){

g.addEdge(i,j,j/i);

}

}

}

This runs in time theta(|E|) which is equal to theta(|V| lg |V|)

}

11. For each of the following applications state which Abstract Data Type (ADT) best fits its implementation: list, stack, queue, binary tree, heap, binary search (or AVL) tree, hash map or graph (directed/undirected, weighted/unweighted). Justify why the ADT you identified is the best one for the application.

[5] a. Web browsers can be configured to keep a cache (i.e. a local copy) of web pages that the user has recently visited. Using such a cache allows the browser to display recently visited pages more quickly. A browser’s cache can easily contain thousands of pages. When the user enters a URL or clicks a link the browser first checks its cache to see if it contains the requested page. If it does, then the page is retrieved from the cache and displayed. If the page is not in the cache then the browser will request it from the web server.

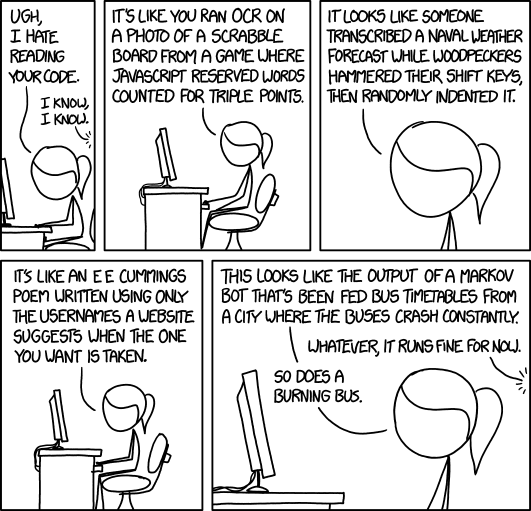
This should be a hashmap. It allows for quick retrieval when given a key. The key should be the url and the value should be an object holding the page.

[5] b. A large office complex requires that employees swipe an ID card when they arrive at work and also when they leave. This allows the company to keep track of the employees who are currently in the complex. Employees are constantly coming and going. Apart from adding and removing employees the most common operations are to determine if a specific employee is in the complex and to generate an alphabetized list of the employees currently in the complex.

This should be a binary search tree, with the employee name as the key. Binary trees allow for fast add and removal, as well as quick determination if an employee is present. Unlike hash maps, the tree also allows for rapid listing of employees in order.

[5] c. The Internet Movie DataBase (IMDB) contains data about virtually every movie ever made, including who acted in each movie. They want to expand their services to be able to quickly answer questions like “list all of the actors/actresses who have ever been in a movie with Clint Eastwood?” or “has Dolly Parton ever been in a movie with Burt Reynolds?”

This should be an undirected graph, with each actor being a vertex. Two vertexs are connected iff those actors have been in a movie together. This graph should be represented as an adjacency list, as the graph is likely to be sparse.



Bonus: Draw all distinct undirected, unweighted graphs with V= {A, B, C}. Two are distinct if one of the graphs contains an edge the other does not.

Bonus: Suppose M is the matrix for an unweighted graph (0 means no edge, 1 means an edge). What does M x M represent?

Bonus: Give a good question that should have been on this test.

Bonus: What are your summer coding plans?