UNIVERSITY OF WATERLOO

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

E&CE 250 – ALGORITHMS AND DATA STRUCTURES

Final Examination Instructor: R.E.Seviora 3 hrs, Dec 18, 2000

Name:				Student id:			Total:
1.	2.	3.	4.	5.	6.	7.	8.

Do all problems. The number in brackets denotes the relative weight of the problem (out of 100). If information appears to be missing from a problem, make a reasonable assumption, state it and proceed. If the space to answer a question is not sufficient, use the last (overflow) page. Only numeric display calculators are allowed. Closed book.

PROBLEM 1 [10]

A. Big Oh

(a) Give the definition of the asymptotic upper bound (big Oh).

(b) For each pair of functions, f(n) and g(n), in the follwing table, state whether f(n)=O(g(n)) and whether g(n)=O(f(n)).

f(n)	*g(n)*	*f(n) = O(g(n)) ?*	*g(n) = O(f(n)) ?*
10n	n^2-10n		
n^3	n^2log n		
ln n	log n		
log (log n)	log n		
2^n	10^n		
n^m	m^n		
cos(nπ/2)	sin(nπ/2)		
n^2	$(n \cos n)^2$		
$10n^3$+3	$5n^3$-6		
5^n	5^{2n}		

B. Induction

Prove by induction that the sum of first n positive odd numbers is n^2: $\sum_{i=1}^{n}(2i-1) = n^2$.

Base Case

Induction Step

PROBLEM 2 [14]

A. Ordered/Sorted Lists

(a) State (in words) the difference between an ordered and a sorted list.

(b) Consider two implementations of sorted list, SortedListAsArray and SortedListAsLinkedList. Give the big Oh running times for the following methods. To answer this question, recall what the basic internal data structure of each of these two implementations is.

method	*SortedListAsArray*	*SortedListAsLinkedList*
void insert (Comparable object)		
boolean isMember (Comparable object)		
Comparable find (Comparable object)		
void withdraw (Comparable object)		
Comparable get (int i)		
Cursor findPosition (Comparable object)		
Cursor.getDatum		
Cursor.withdraw		

B. Visitors

Consider a container that contains only instances of the `Integer` class. The `Integer` class is a wrapper for the primitive type `int`; its method `public int intValue()` returns the value of the integer wrapped. Write the code for a `MaxIntVisitor` that finds the maximum value of all `Int's` in the container. Write also the accessor `getMaxInt` and returns the maximum value of all integers in the container. (Naturally, an instance of `MaxIntVisitor` must have 'visited' all objects in the container `for getMaxInt` to report correct value.)

```
public class MaxIntVisitor
 extends AbstractVisitor {
 //fields



 //methods
 public void Visit (Object obj) {









 public boolean isDone() {




 public int getMaxInt () {
```

PROBLEM 3 [14]

A. Hashing

(a) Consider a hash table with separate chaining with ten hash locations. Using the hash function of h(x)=x mod 10, insert the keys 33, 54, 69, 74, 18, 19 (in the order given) into the hash table. Draw the resulting hash table. (Note that to keep this example simple, we use table size which is not prime.)

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

(b) Open addressing scatter tables: state the two basic properties a collision resolution function should satisfy.

(c) Consider an open addressing scatter table with ten slots. For the hash function h(x)=x mod 10, insert the keys 33, 54, 69, 74, 18, 19 (in the order given) into the table. Use *linear probing* for collision resolution. Draw the result in the table (c) below.

(d) Consider an open addressing scatter table with ten slots. Using the hash function h(x) =x mod 10, insert the keys 33, 54, 69, 74, 18, 19 (in the order given) into the table. Use *quadratic probing* for collision resolution. Draw the result in the table (d) below.

(e) Consider an open addressing scatter table with ten slots. Using the hash function h(x)=x mod 10, insert the keys 33, 54, 69, 74, 18, 19 (in the order given) into the hash table. Use the *secondary hash function* h'(x) = 1 + (x mod 9) for collision resolution. Draw the result in the table (e) below.

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

(c)

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

(d)

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

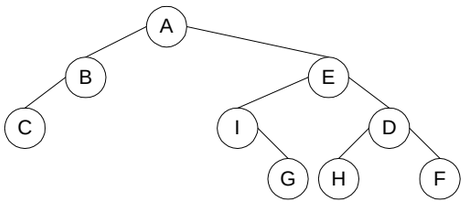
(e)

(f) Consider your answer to (d) above. What happens when you attempt to insert the key 94 into the final hash table obtained after inserting the given keys. Explain.

PROBLEM 4 [12]

A. Tree Traversal

Consider the binary tree shown immediately below. For each of the traversals listed, give the order in which the nodes are visited.



preorder									
inorder									
postorder									
breadth-first									

B. Binary Tree

Consider a binary tree whose nodes contain single-character keys. When the data fields of the nodes are output inorder, the output is `A B C D E F G H I J`, and when they are output in postorder, the output is `B C D A E H G J I F`. Draw the binary tree.

C. Tree Relationships

(a) What is the relationship between the height of the tree and the number of internal nodes for

(i) perfect binary tree

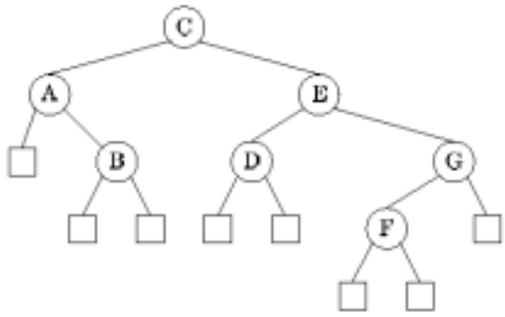
(ii) complete binary tree

(b) Derive an expression of the total space needed to represent a binary tree of n internal nodes. Consider the course text setting, in which `BinaryTree` extends `AbstractContainer`. Assume four-byte integers and object references.

PROBLEM 5 [14]

A. AVL balance trees

(a) For each node shown in the tree below, show its depth, height, and AVL balance factor. Write your answers in the table.



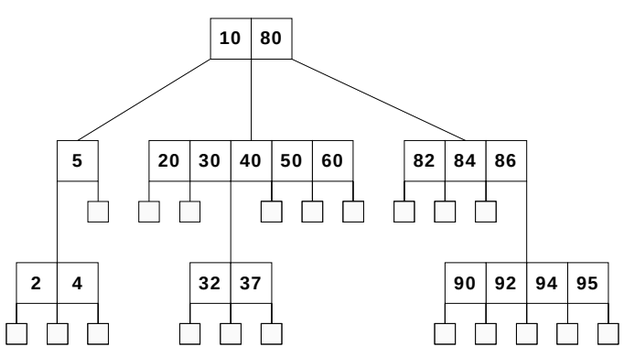
node	*depth*	*height*	*bal fact*
A			
B			
C			
D			
E			
F			
G			

(b) Draw the sequence of AVL trees obtained when the following keys are inserted one-by-one, in the order given into an initially empty AVL search tree: {F; E; A; B; D; C; G}. Identify the rotations, if any.

B. M-way and B-Trees

(a) An M-way search tree is called a B-tree if three conditions are satisfied. List these conditions.

(b) Consider the 6-way M-tree below. Show the tree after an item with key 70 is inserted. Only the modified parts need to be shown.



PROBLEM 6 [8]

A. Binary Heap

(a) Consider an array implementation of a binary min heap in which all of the keys are single letters. The heap is initially empty. The following items are inserted, one-by-one, in the following order: {D, C, A, G, B, F, E}. Complete the table below by showing the array contents after each insertion.

item	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]
D							
C							
A							
G							
B							
F							
E							

(b) Consider the binary min heap given in the first line of the table below. Show the contents of the array after the operation `dequeMin()`. Show intermediate steps in the execution of `dequeMin()`.

	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]
initial	B	C	E	F	D	G	H
dequeMin - step 1							

PROBLEM 7 [16]

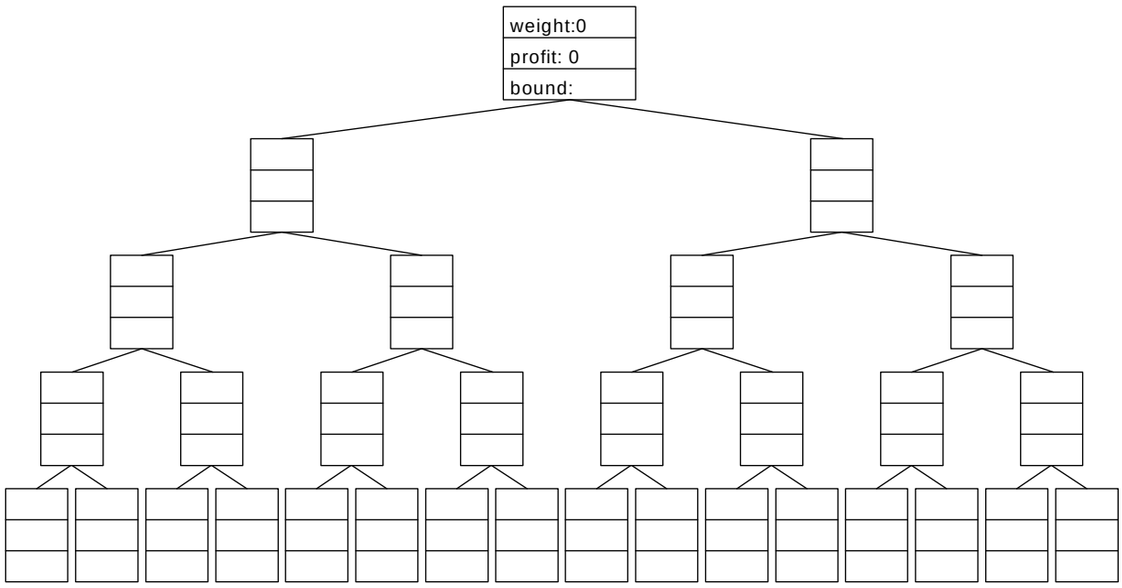
A. Branch-and-Bound Backtracking Strategy

One of the 'standard' problems in the discussion of algorithms is the 0/1 knapsack problem. A backtracking algorithm could be devised for the solution of this problem. The performance of such algorithm could be further improved by devising a suitable bounding function and using it in the solution process to reduce the number of nodes considered.

(a) Define a suitable bounding function and show it below.

(b) Consider the case when four items with the weights and profits shown below are available. The total carrying capacity of the knapsack is 100. Show the solution space for this problem using the tree template given. If needed, add other nodes to the tree. Mark each node with the total weight, the profit accumulated to that point, and the bound value. Cross out the nodes which need not be explored. (Observation: establish first the order in which items are considered for inclusion in the backpack.)

item	weight	profit	prof density
1	10	15	1.5
2	50	35	0.7
3	35	14	0.4
4	70	42	0.6



B. CPU Scheduling Problem

At time T_0, a single computer receives N tasks $\{t_1,t_2,..,t_N\}$ to execute. Only one task can be executed at a time and a task must execute to completion. The times required to execute each task are known: $\{e_1,e_2,..,e_N\}$. The term 'the time a task t_i spends in the system' is defined as the sum of the time the task waits for its execution to start (w_i) plus its execution time e_i.

The CPU scheduling problem requires that the execution of the tasks $\{t_1,t_2,..,t_N\}$ be scheduled, i.e. the order of their execution defined, at T_0 in such a way that the average time a task spends in the system is minimized.

(i) Give a mathematical formulation of this problem, i.e. specify the objective function, the constraints, plus others if needed.

(ii) Develop a brute force solution to the CPU scheduling problem for the case of three tasks, with e_1=3, e_2=10, and e_3=6 time units. Identify the solution(s).

Task Exec Order	*Total Time Spent in the System(all tasks)*	*Avg Task Time In Syst*

(iii) Describe a greedy strategy for solution to task scheduling. Indicate what is the strategy greedy by.

(iv) Your answer to (iii) should give the lowest average time spent in the system for the set $\{t_1,t_2,..,t_N\}$. Present an argument why this is the case with your answer.

PROBLEM 8 [12]

A. Median-of-three quicksort.

(a) Outline how the median-of-three method selects the pivot for quicksort. Use text/diagram, not Java code.

(b) Trace the execution of the median-of-three quicksort algorithm as it sorts the sequence given in the first row of Figure P7A. Assume that the cutoff value is 2. Subarrays of length <= 2 are sorted by straight insertion sort. Show the intermediate steps.

B. Radix sort.

(a) Trace the execution of the radix sort algorithm as it sorts the sequence given in the first row of Figure P7B.

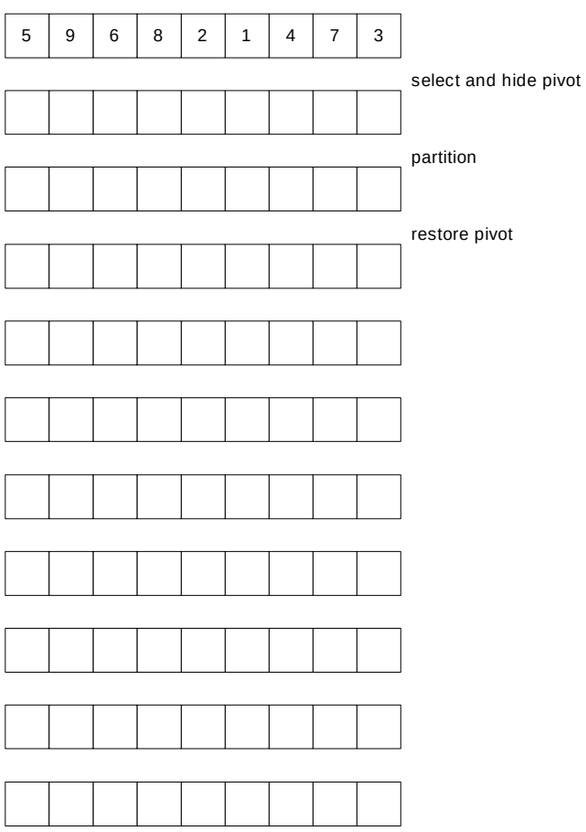


Fig P7A

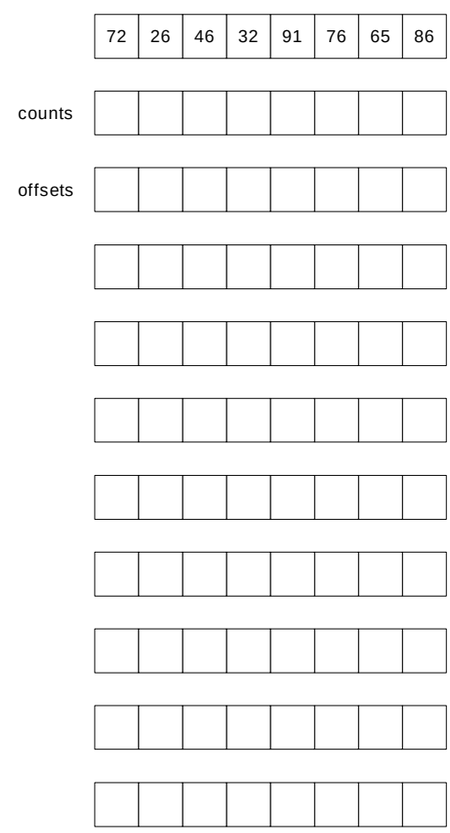


Fig P7B

OVERFLOW SHEET [Please identify the question(s) being answered.]