Yao-Wen Chang, Hui-Ru Jiang, and Chien-Mo Li

EE4033; 901/39000: Midterm #1

1	2	3	4	5	6	7	8	9	10	Total
/10	/10	/15	/10	/10	/10	/14	/8	/5	/13	/105

Name	Student ID	Class
		Y

Instructions.

1. This is a 170-minute test, with a total of 105 points available.
2. There are 14 pages and 10 problems.
3. You are allowed to refer one A4-size cheat sheet with only hand-written notes.
4. Note that you do NOT need to give the pseudo code of the algorithms listed in the lecture notes once referenced; instead, you may simply give the name of the algorithm.
5. Please hand in your cheat sheet and problem sheets with your solutions for avoiding penalty.
6. **Pace Yourself and Good Luck!**

Problem 1. (10 pts total)
Prove or disprove the following two statements. If the statement is true, provide a proof. If the statement is false, give a counterexample. No partial credit will be given if no proof or counterexample is provided.

(a) (5 pts) For any two positive functions $f(n)$ and $g(n)$, either $f(n) = O(g(n))$ or $g(n) = O(f(n))$.

(b) (5 pts) Let $f_1(n), f_2(n), \cdots, f_i(n), \cdots$ be an infinite series of functions such that $f_i(n) = O(n)$ for every i. Let $g(j) = f_j(j)$ for all positive integers j. Then, we have $g(n) = O(n)$.

Problem 2. (10 pts total)
Solve the following recurrences. You only need to obtain the asymptotic solution in $\Theta()$ notation. If you use the master theorem, please specify all parameters and briefly verify all conditions. Assume $T(n)$ is $\Theta(1)$ for $n \leq 2$.

(a) (5 pts) $T(n) = 4T(n/2) + n^2 + n$ for all n that is a power of 2. Show your steps.

(b) (5 pts) $T(n) = T(\alpha n) + T(\beta n) + n$, where $\alpha + \beta < 1$. Derive $\Theta()$ based on $O()$ and $\Omega()$ and show your steps.

Problem 3. (15 pts total) Algorithm design.

A city's skyline is the outer contour of the silhouette formed by all the buildings in that city when viewed from a distance. Given n rectangular buildings in a 2-dimensional city, compute the skyline of these buildings,

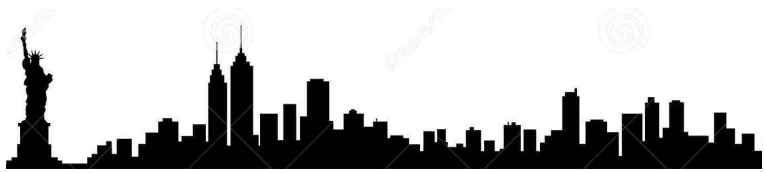
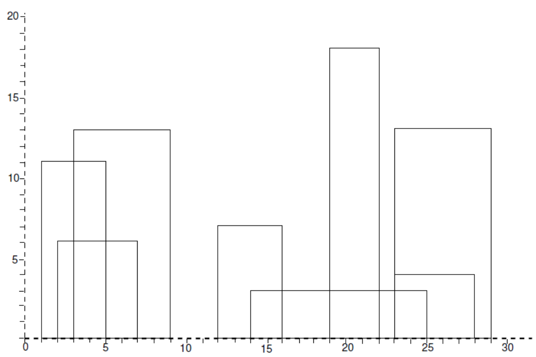
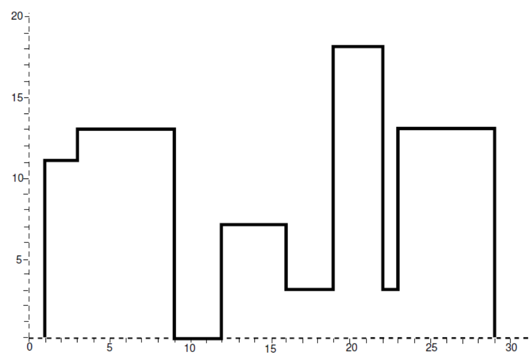


Figure 1: New York Skyline

eliminating hidden lines. All buildings share common bottom (absolutely flat surface at height 0), and every building B_i ($1 \leq i \leq n$) is represented by triplet (L_i, H_i, R_i), where L_i and R_i denote the x coordinates of the left and right of the ith building, and H_i denotes its height. A skyline of a set of n buildings is a list of x coordinates and the heights connecting them arranged in order from left to right. Example: The skyline of the buildings
$\{(3, 13, 9), (1, 11, 5), (12, 7, 16), (14, 3, 25), (19, 18, 22), (2, 6, 7), (23, 13, 29), (23, 4, 28)\}$
is
$\{(1, 11), (3, 13), (9, 0), (12, 7), (16, 3), (19, 18), (22, 3), (23, 13), (29, 0)\}$.
Please note that there must be no consecutive horizontal lines of equal height in the output skyline.



buildings



skyline

(a) (5 pts) Let the size of a skyline be the total number of elements (coordinates and heights) in its list. Design an algorithm for combining a skyline A of size n_1 and a skyline B of size n_2 into one skyline S of size $O(n_1 + n_2)$. Your algorithm should run in time $O(n_1 + n_2)$. Justify the running time of your algorithm.

(b) (5 pts) Design an $O(n \lg n)$ algorithm for finding the skyline of n buildings. Any algorithm that requires $\Omega(n^2)$ running time will not receive any credit. Justify the running time of your algorithm. **Attention: You may assume the algorithm of (a) is available when you design the algorithm of (b).**

(c) (5 pts) Briefly justify the correctness of part (a) and part (b). [Hint: Loop invariant; mathematical induction]

Problem 4. (10 pts total) New Partition.

Harry claims he invented a new partition algorithm shown as follows, where p and r are the smallest and largest index of array A. The pivot element (x) is initially $A[p]$. Please note that the input array size $= n$.

```
H-Partition (A, p, r)
x = A[p]
i = p - 1
j = r + 1
while TRUE
    repeat
        j = j - 1
    until A[j] ≤ x
    repeat
        i = i + 1
    until A[i] ≥ x
    if i < j
        exchange A[i] and A[j]
    else return j
```

(a) (2 pts) Demonstrate the operation of H-Partition on the array $A =< 11, 2, 15, 6, 2, 17, 3, 22 >$. Suppose we call H-Partition$(A, 1, 8)$. Show the array after each iteration of the while loop and the values of variables i and j.

Iteration	A	i	j
1			
2			
3			
	return j = ?		

(b) (3 pts) What is the time complexity of H-Partition? Please use correct Θ notation.

(c) (2 pts) Please write a pseudo code for *Quicksort_H* algorithm using the H-Partition.

(d) (3 pts) Is your *Quicksort_H* a stable sorter? Please explain your answer.

Problem 5. (10 pts total) Heap.
Give the array $A[i]$

i	1	2	3	4	5	6	7	8	9	10	11
$A[i]$	7	1	5	11	10	14	3	9	8	2	13

(a) (5 pts) Build a max heap using the BUILD-MAX-HEAP algorithm in the textbook. Show your work step-by-step. Please show your answer in the following array format.

index	1	2	3	4	5	6	7	8	9	10	11
step1	7	1	5	11	10	14	3	9	8	2	13
step2											
...											

(b) (3 pts) Describe an algorithm to decrease a key in the max heap.

(c) (2 pts) For the following given heap, use your algorithm in part (b) to decrease the key of $A[2]$ to 6. Show the resulting heap step-by-step.

index	1	2	3	4	5	6	7	8	9	10	11
step1	100	90	80	70	65	60	55	1	2	3	5
step2											
...											

Problem 6. (10 pts total) Order Statistics.
Given n numbers that are distributed between zero and one. Given the original Bucket-sort algorithm in the textbook.

```
Bucket-Sort (A)
  n = A.length
  Let B[0..n − 1] be a new array
  for i = 0 → n − 1
      make B[i] an empty list
  for i = 0 → n
      insert A[i] to list B[⌊nA[i]⌋]
  for i = 0 → n − 1
      sort list B[i] with insertion sort
  concatenate the lists B[0], · · · , B[n − 1] together in order
```

(a) (5 pts) Describe an order statistic algorithm (*i.e.* find the ith smallest element of n elements) by modifying the Bucket-Sort algorithm. Your proposed algorithm must be faster than the original Bucket-Sort. Just describe your idea in words, no pseudo code is needed.

(b) (3 pts) What is the worst-case running time of your order statistics? Please explain.

(c) (2 pts) Assume the numbers are uniformly distributed. What is the expected running time of your order statistics? Please explain.

Problem 7. (14 pts total) Search trees.

(a) (3 pts) Give the binary search tree that results from successively inserting the keys 3, 8, 2, 9, 6, 5, 1, 7 into an initially empty tree.

(b) (3 pts) Label each node in the tree with R or B denoting the respective colors RED and BLACK so that the tree is a legal red-black tree.

(c) (4 pts) Give the red-black tree that results from inserting the key 4 into the tree of (b).

(d) (4 pts) Give the red-black tree that results from deleting the key 9 also from **the tree of (b) (NOT from the tree of (c)!!)**.

Attention: There is only one correct answer to (a) and (b) each. You should work on (a) and (b) very carefully because your correctness for (c) and (d) also depends on that of (a) and (b)!!

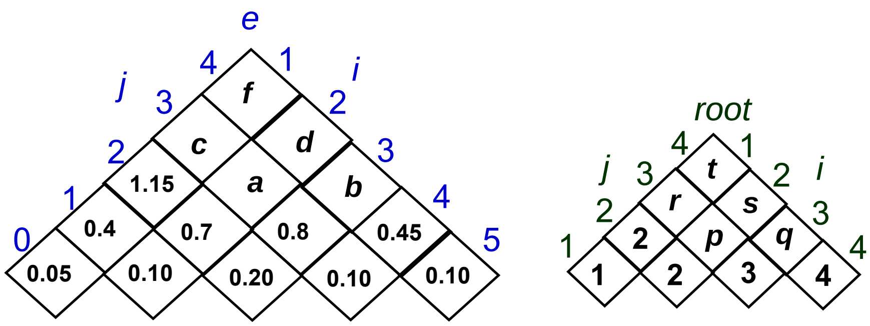
Problem 8. (8 pts total)
You are asked to determine the cost and structure of an optimal binary search tree for a set $K =< k_1, k_2, k_3, k_4 >$ of $n = 4$ keys with the following probabilities:

i	0	1	2	3	4
p_i	-	0.10	0.10	0.20	0.05
q_i	0.05	0.10	0.20	0.10	0.10

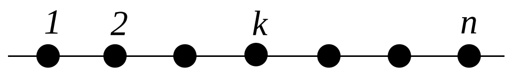
a set of probabilities $P =< p_1, p_2, p_3, p_4 >$ for searching the keys in K and $Q =< q_0, q_1, q_2, q_3, q_4 >$ for unsuccessful searches, as discussed in class.

(a) (6 pts) Find a, b, c, d, and f in the e table (5 pts), and p, q, r, s, and t in the *root* one (1 pt), where $e[i, j]$ gives the expected cost of searching an optimal binary tree containing the keys $k_i, \ldots, k_j$, and $root[i, j]$ records the root of the subtree containing the keys $k_i, \ldots, k_j$.

(b) (2 pts) Find an optimal binary search tree of the given probabilities and give the expected search cost of the tree.



Problem 9. (5 pts total)
Label the points of the below boundary $1, 2, \ldots, n$ starting from the left and to the right. Let $N(i, j)$ denote the maximum number of non-intersecting connections between points i and j that can be connected by using the upper side of the boundary. If there already exists a connection between points 1 and k, compute the maximum number of non-intersecting connections in the upper side of the boundary, $N(1, n)$, in terms of the 2-argument function N with arguments $k-1, k, k+1$ and some constant(s).



Problem 10. (13 pts total)

Suppose that you are given three strings of characters: $X = x_1x_2 \dots x_m$, $Y = y_1y_2 \dots y_n$, and $Z = z_1z_2 \dots z_{m+n}$. Z is said to be a *shuffle* of X and Y if Z can be formed by interspersing the characters from X and Y in a way that maintains the left-to-right ordering of the characters from each string. For example, *cchocohilaptes* is a shuffle of *chocolate* and *chips*, but *chocochilatspe* is not.

(a) (7 pts) Give the necessary and sufficient condition for the logic query $Z_{i+j} ==$ *shuffle*(X_i, Y_j), $\forall i, j, 1 \le i \le m, 1 \le j \le n$, in terms of $Z_{i+j-1}, X_{i-1}, Y_{j-1}, z_{i+j}, x_i$, and y_j (i.e., the optimal substructure for this problem). Recall what we did for the LCS problem discussed in class.

(b) (4 pts) Design a dynamic programming algorithm Is_Shuffle that takes as inputs X, Y, Z, m, and n, and determines whether Z is a shuffle of X and Y, where Algorithm Is_Shuffle should use table s of Boolean entries to store the results: $s[i, j]$ is TRUE if and only if $Z_{i+j} =$ *shuffle*(X_i, Y_j), $\forall i, j, 1 \le i \le m, 1 \le j \le n$.

(c) (2 pts) What is the running time of your algorithm?

If you need more space to write down your answers, you can use this page. Please specify the problem number!