Note: This exam has 5 questions for a total of 100 points. Answer all questions. Write all your answers on the blue books.

Question I (25 points total)


(b) Let $n \geq 1$ be an integer. Suppose a trial consists of choosing an integer $y$ uniformly randomly in the range $[1..3n]$. Assume that successive trials are independent and that we carry out a total of $m$ trials. If a trial produces an integer $y$ which is divisible by 3 (i.e., $y$ leaves a remainder of zero when divided by 3), then we insert the integer $y$ into a linked list $L$. ($L$ is empty before we start the trials.) Let $X$ denote the random variable that represents the number of values in $L$. Compute (i) the expectation and (ii) the variance of $X$. Show work. (15 points)

Question II (25 points total)

(a) A file contains a string made up of six symbols $U$, $V$, $W$, $X$, $Y$ and $Z$, with respective frequencies 10, 10, 15, 15, 20 and 30. Construct an optimal binary prefix code for the symbols using Huffman’s algorithm. Your answer must show the tree constructed as well as the prefix codes for the six symbols. There is no need to show the intermediate stages. (13 points)

(b) In the bin packing problem, we are given a collection of $n$ items with sizes $s_1, s_2, \ldots, s_n$, where $0 < s_i \leq 1$ for $1 \leq i \leq n$. We are also given bins each with capacity 1. Assume that the bins are numbered 1, 2, \ldots, etc. The goal is to pack all the given items into a minimum number of bins such that the total size of all the items in each bin is at most 1. This problem is known to be NP-complete. So, heuristics are used in practice. One such heuristic, called First Fit, is the following.

(I) Sort the items in non-decreasing order of size. Without loss of generality, let $s_1 \leq s_2 \leq \ldots \leq s_n$ denote the sorted order. All the bins are initially empty.

(II) for $i = 1$ to $n$ do

(i) Consider the bins in the order 1, 2, etc. and find the first bin where the item with size $s_i$ fits. (This may entail using a bin that is currently empty. For example, item $s_1$ will be added to bin 1.)

(ii) Add item with size $s_i$ to the chosen bin.

(III) Output the number of nonempty bins.

Give an example to show that First Fit will not produce optimal solutions in general.
Your answer must clearly specify the sizes of the items, the optimal number of bins for the set, and the number of bins used by the heuristic along with an explanation of why the heuristic uses the specified number of bins. (12 points)

**Question III** (15 points total)

You are given a sequence $S$ of $n$ distinct integers to sort, where $n = r^3$ for some positive integer $r$. This input sequence $S$ consists of $r$ subsequences, $S_1$, $S_2$, ..., $S_r$, with each subsequence containing exactly $r^2$ elements. For each subsequence $S_i$, the elements in $S_i$ are all larger than any element in $S_{i-1}$ and are all smaller than any element in $S_{i+1}$, $2 \leq i \leq r - 1$. In addition, all the elements in $S_1$ are smaller than any element in $S_r$ and all the elements in $S_r$ are larger than any element in $S_1$.

(a) Prove that the number of possible sorted orders for the sequence $S$ is $\left[(r^2)!\right]^r$. (7 points)

(b) Use the result in part (a) to show that to sort the given input, any comparison-based sorting algorithm must use $\Omega(n \log n)$ comparisons. (8 points)

**Question IV** (15 points)

Consider the following problem:

**Given:** A set $S = \{x_1, x_2, \ldots, x_n\}$ of positive integers, a positive integer $k \leq n$ and another positive integer $Y$. The integers in $S$ are all distinct.

**Requirement:** Determine whether there is a subset $S'$ of $S$ such that $|S'| = k$ and the sum of the elements in $S'$ is at most $Y$. If the answer is yes, then output one such subset $S'$.

You are required to present an algorithm for the above problem. To be considered for full credit, the running time of your algorithm must be $O(n)$. You can get up to 8 points if the running time of your algorithm is $O(n \log n)$ instead of $O(n)$. No credit will be given for any algorithm whose running time is worse than $O(n \log n)$.

Your answer must include a clear description of the algorithm in the form of high-level pseudocode, a clear explanation of why your algorithm is correct and an analysis of its running time.
**Question V** (20 points total)

Given an undirected graph $G$ with node set $V$ and edge set $E$, an independent set of $G$ is a subset $V' \subseteq V$ of nodes such that there is no edge in $E$ between any pair of nodes in $V'$. A maximum independent set for $G$ is an independent set of maximum cardinality.

**Example:** In the following graph, both $\{v_1, v_4\}$ and $\{v_2\}$ are independent sets. But $\{v_1, v_3, v_4\}$ is not an independent set because of the edge $\{v_1, v_3\}$. The size of a maximum independent set for this graph is 2.

This problem explores a dynamic programming approach to compute the size of a maximum independent set in a full binary tree $T$ (i.e., a binary tree in which each internal node has exactly two children). Assume that $T$ has $n$ nodes and that the nodes have been numbered 1, 2, 3, ..., $n$, so that for each internal node numbered $i$, the children of $i$ have numbers less than $i$.

For each node $i$, our dynamic programming algorithm maintains two values, denoted by $X[i]$ and $Y[i]$. Here, $X[i]$ denotes the size of a maximum independent set for the subtree rooted at $i$ under the condition that node $i$ is not included in the maximum independent set. Similarly, $Y[i]$ denotes the size of a maximum independent set for the subtree rooted at $i$ under the condition that node $i$ is included in the maximum independent set.

(a) Suppose the node numbered $i$ is a leaf. Indicate how $X[i]$ and $Y[i]$ can be computed. Also, estimate the running time for this computation and indicate how you arrived at the estimate. (4 points)

(b) Suppose the node numbered $i$ is an internal (i.e., non-leaf) node and the left and right children of $i$ are numbered $j$ and $k$ respectively. Indicate how $X[i]$ and $Y[i]$ can be computed using the values of $X[j]$, $Y[j]$, $X[k]$ and $Y[k]$. Also, estimate the running time for this computation and indicate how you arrived at the estimate. (8 points)

(c) Estimate the time needed to compute all the entries of the $X$ and $Y$ arrays. Be sure to indicate how you arrived at the estimate. (4 points)

(d) Suppose we have the values for all the entries of the arrays $X$ and $Y$. Indicate how the size of a maximum independent set for $T$ can be computed. Also, estimate the running time for this computation and indicate how you arrived at the estimate. (4 points)
Pseudocode for Bucket-Sort:

Bucket-Sort(A)

// Each element of A[1..n] is a real number in the interval [0,1).
// B[0 .. n-1] is an array of pointers.

1. for i = 1 to n do
   Insert A[i] into the list pointed to by B[floor(n*A[i])].

2. for i = 0 to n-1 do
   Sort list B[i] into increasing order using Insertion-Sort.

3. Concatenate lists B[0], B[1], ..., B[n-1] together in order.

Note: In Step 1 above, we have used floor(n*A[i]) to denote ⌊n * A[i]⌋.