Question I

Part (a): After Step 1 of Bucket-Sort, $B[0]$ points to the node with value 0.11, $B[1]$ points to a list with two nodes, one containing the value value 0.2 and the other containing 0.31, $B[2]$ points to the node with value 0.42, $B[3]$ is NULL and $B[4]$ points to the node with value 0.9.

Part (b): Let a trial be called a ‘success’ if the value $y$ is divisible by 3. In the range $[1..3n]$, there are $n$ integers that are divisible by 3. Since the choice is uniformly random, the probability $p$ of success is $1/3$. Since the successive trials are independent and the random variable $X$ represents the number of successes in $m$ trials, $X$ has a binomial distribution.

For a binomial distribution with $m$ trials and success probability $p$, the expected number is $mp$. Therefore, the expected size of list $L = m/3$.

For a binomial distribution with $m$ trials and success probability $p$, the variance is $mp(1-p)$. Therefore, the variance of the size of list $L = 2m/9$.

Question II

Part (a):

<table>
<thead>
<tr>
<th>Character</th>
<th>Prefix code</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>000</td>
</tr>
<tr>
<td>V</td>
<td>001</td>
</tr>
<tr>
<td>W</td>
<td>100</td>
</tr>
<tr>
<td>X</td>
<td>101</td>
</tr>
<tr>
<td>Y</td>
<td>01</td>
</tr>
<tr>
<td>Z</td>
<td>11</td>
</tr>
</tbody>
</table>
Part (b): Consider 4 items $I_1, I_2, I_3$ and $I_4$ of sizes $1/3, 1/3, 2/3$ and $2/3$ respectively. (The sizes are already in sorted order.)

An optimal packing puts items $I_1$ and $I_3$ in one bin and items $I_2$ and $I_4$ in another bin, for a total of 2 bins.

The First Fit approach would put items $I_1$ and $I_2$ into the same bin, and so must use two more bins for the remaining two items each of which has size $2/3$. So, the number of bins used by First Fit is 3, which is not optimal.

Note: It can be shown that for any input, the number of bins used by First Fit is at most twice the optimal number of bins.

---

Question III

Part (a): Each subsequence $S_i$ contains $r^2$ elements, $1 \leq i \leq r$. Thus, the number of possible permutations of the elements in any subsequence is $(r^2)!$. From the statement of the problem, it can be seen that any sorted order of the sequence $S$ must consist of the elements of $S_1$ (in sorted order), followed by the elements of $S_2$ (in sorted order), ..., followed by the elements of $S_r$ (in sorted order). Since there are $(r^2)!$ possible permutations for each subsequence, the total number of possible sorted orders is $[(r^2)!]^r$.

---

Part (b): Consider any comparison-based sorting algorithm $A$ for the sequence $S$. From the result of Part (a), the decision tree representation of $A$ must have $[(r^2)!]^r$ leaves (since any of them may be the true sorted order). Thus, the height of the decision tree, that is, the number of comparisons made by $A$ in the worst-case is at least $\log_2 [(r^2)!]^r = r \log_2 (r^2)! = \Omega(r \times r^2 \times \log_2 (r^2)) = \Omega(r^3 \log_2 r)$. Now, $r^3 = n$ and $\log_2 r = \log_2 (n^{1/3}) = \frac{1}{3} \log_2 n$. Hence, the lower bound $\Omega(r^3 \log_2 r) = \Omega(n \log n)$.

---

Question IV

The following is a simple observation about the problem.

Observation: Let $S'$ be the subset consisting of the $k$ smallest integers in $S$. There is a solution to the problem if and only if the sum of the values in $S'$ is at most $Y$.

Algorithm I: (Running time: $O(n \log n)$)

1. Sort the set $S$ into increasing order using Heap-Sort.

2. Let $S'$ consist of the first $k$ elements in the sorted order.

3. If the sum of the values in $S'$ is at most $Y$, output the message “There is a solution” and the set $S'$; otherwise, output the message “There is no solution”.

Correctness: This is a direct consequence of the above observation.

Running time: Note that Step 1 uses $O(n \log n)$ time while Steps 2 and 3 use only $O(n)$ time. Thus, running time of Algorithm I is $O(n \log n)$.
Algorithm II: (Running time: $O(n)$)

1. Find the $k^{th}$ smallest value in $S$ using the linear time selection algorithm. Let $y$ denote this value.

2. Let $S'$ consist of all the elements of $S$ which are $\leq y$.

3. If the sum of the values in $S'$ is at most $Y$, output the message “There is a solution” and the set $S'$; otherwise, output the message “There is no solution”.

Correctness: Since the elements of $S$ are distinct, the size of subset $S'$ chosen in Step 2 is equal to $k$. In other words, $S'$ contains the $k$ smallest values of $S$. Now, the correctness of Algorithm II is a direct consequence of the above observation.

Running time: There are three steps in the algorithm and each step uses only $O(n)$ time. Thus, running time of Algorithm II is $O(n)$.

Question V:

Part (a): For a leaf node $i$, the subtree rooted at $i$ consists of just that node. Therefore, $X[i] = 0$ and $Y[i] = 1$. Since these are two simple assignment statements, the time need to compute $X[i]$ and $Y[i]$ is $O(1)$.

Part (b): Given that $i$ is an internal node whose left and right children are $j$ and $k$, we can compute $X[i]$ and $Y[i]$ as follows.

(i) In computing $X[i]$, we note that $i$ must be excluded from the independent set. Therefore, we are free to choose or not choose $j$ and $k$ in the independent set. Therefore,

$$X[i] = \max \{X[j], Y[j]\} + \max \{X[k], Y[k]\}.$$ 

(ii) In computing $Y[i]$, we note that $i$ must be included in the independent set. Therefore, we cannot choose either $j$ or $k$ in the independent set. Therefore,

$$Y[i] = 1 + X[j] + X[k].$$

The computation of $X[i]$ uses $O(1)$ time since it involves the sum of two numbers, each of which is the larger of two other numbers. The computation of $Y[i]$ also uses $O(1)$ time since it involves just the sum of three numbers. other numbers.

Part (c): Since each entry of $X$ and $Y$ can be computed in $O(1)$ time and the total number of entries is $2n$, the time needed to compute all the entries of $X$ and $Y$ is $O(n)$.

Part (d): Note that the root is numbered $n$. Therefore, after computing all the entries of $X$ and $Y$, the size of a maximum independent set can be computed as $\max \{X[n], Y[n]\}$. Since this involves finding the larger of two numbers, the time needed for the computation is $O(1)$.