I (10 points)
Note that the optimal substructure of LCS holds. We introduce another variable to record
the ending symbol of the common subsequence. Let $c[i, j, a]$ be the length of the longest
restricted common subsequence (LRCS) between two strings $x_1 x_2 \cdots x_i$ and $y_1 y_2 \cdots y_j$ ending
with symbol $a$. Let $S$ be the dictionary of alphabets and $s$ be its size. Since the LRCS requires
that no two consecutive symbols are equal, the recursion formula becomes:

$$
c[i, j, a] = \begin{cases} 
0 & \text{if } i = 0 \text{ or } j = 0, \\
c[i-1, j-1, a] & \text{if } i, j > 0 \text{ and } a \notin \{x_i, y_j\}, \\
c[i, j-1, a] & \text{if } i, j > 0 \text{ and } a = x_i \text{ and } a \neq y_j, \\
c[i-1, j, a] & \text{if } i, j > 0 \text{ and } a = y_j \text{ and } a \neq x_i, \\
\max_{b \neq a}\{c[i-1, j-1, a], c[i-1, j-1, b] + 1\} & \text{if } i, j > 0 \text{ and } a = y_j = x_i.
\end{cases}
$$

Given the strings $X = x_1 x_2 \cdots x_m$ and $Y = y_1 y_2 \cdots y_n$, the maximum length will be determined
by $\max_a\{c[m, n, a]\}$.

Note that in the fifth case the maximization can be reused for all $b \in S$ (except that $a$
eq b that is maximum solution, in which the second largest solution is chosen). Hence
algorithm runs in $O(mns)$.

II (10 points)
Let $T$ denote the 2-4 tree and $k$ be the key to insert into $T$. The insertion is executed as:

1. From root($T$) search downwardly to locate the leaf node $x$ to be inserted into.
2. Insert $k$ into $x$.
3. If $x$ has 4 keys, say $\{k_1, k_2, k_3, k_4\}$, repeat:
   i. If $x$ is the root, create new node as the root, move $k_3$ to the root node, and split the
      rest of $x$ into two nodes $x_1$ and $x_2$ such that $x_1$ contains $\{k_1, k_2\}$ and $x_2$ contains $\{k_4\}$.
      Let the two children of the root point to $x_1$ and $x_2$ respectively. Increase the height of $T$
      by 1, return $T$.
   ii. Else, randomly select $k_2$ or $k_3$, wlog let us take $k_3$ for example, insert $k_3$ into $x$’s parent
      node, denoted by $y$, and split the rest of $x$ into two nodes $x_1$ and $x_2$ such that $x_1$ contains
      $\{k_1, k_2\}$ and $x_2$ contains $\{k_4\}$. Update the two pointers in $y$ before $k_3$ and after $k_3$ to
      point to $x_2$ and $x_3$, respectively, then update $x$ by $y$.

   till $x$ has less than 4 keys.
4 Return $T$.

Step 1 takes $O(\log n)$ time. Step 2 takes $O(1)$ time. Step 3 starts from a leaf node and runs at most to the root, therefore it takes $O(\log n)$ time. In sum the algorithm runs in $O(\log n)$. 