HOMEWORK 8

8.1: Actually any MST must contain the second smallest edge $e$. We prove that by contradiction. Assume there exists an MST $T$ which does not contain $e$. Adding $e$ to the MST forms a cycle $p$. The maximum-weight edge $e'$ in $p$ must be heavier than $e$ because a cycle consists of three edges at least, and $e$ is the second smallest edge in $G$. Then breaking the cycle $p$ by taking off $e'$ will create another spanning tree $T'$ with less weight than $T$, which is an MST by assumption. From the argument, we can also see that the statement does not hold for the third smallest edge.

8.2: The shortest-path tree rooted at $A$ is $\{AB, AC\}$, while the minimum spanning tree is $\{AB, BC\}$.
8.3: Move $P$ to the position where the characters of $T$ and $P$ are unmatched, instead of just moving ahead by 1 each time.

\[
\text{NAIVE-PATTERN-MATCHER}(T, P) \\
n \leftarrow \text{length}[T] \\
m \leftarrow \text{length}[P] \\
s \leftarrow 0 \\
\text{while } s \leq n - m \\
\quad j \leftarrow 1 \\
\quad \text{while } j \leq m \text{ and } P[j] = T[s + j] \\
\quad \quad j \leftarrow j + 1 \\
\quad \text{if } j > m \\
\quad \quad \text{then print "Pattern occurs at shift" } s \\
\quad s \leftarrow s + \max(1, j - 1)
\]

8.4: In order to prove that $B_{n \times n} = A_{n \times m}^T A_{m \times n}$ is symmetric, we need to show that $B[i, j] = B[j, i]$, for any $i, j \in \{1, 2, \ldots, n\}$. By the definition of matrix multiplication, we have

\[
B[i, j] = \sum_{k=1}^{m} A[i, k] A[k, j] = \sum_{k=1}^{m} A[k, i] A[k, j], \quad \text{and} \\
B[j, i] = \sum_{k=1}^{m} A[j, k] A[k, i] = \sum_{k=1}^{m} A[k, j] A[k, i]
\]

8.5: Append the following piece of codes to MAXPROB. The running time for this operation is $O(n)$.

\[
i \leftarrow 1 \\
\text{while } i \leq n \\
\quad \text{print } q[i] \\
\quad \text{print SPACE} \\
\quad i \leftarrow i + |q[i]|
\]

8.6: The basic idea is scanning the sequence $A$ of $n$ numbers, and appending the current number to the longest monotonically increasing subsequence before it whose last element is smaller than the number. We use auxiliary arrays $S1[1..n], S2[1..n]$, so that $S1[i]$ points to the last element of the $i$-size candidate subsequence currently maintained, and $S2[i]$ points to the element right before $A[i]$ in the same subsequence. $L$ maintains the longest length.

\[
\text{FIND-LMIS}(A, n) \\
\quad \text{for } i \leftarrow 1 \text{ to } n \\
\quad \quad S1[i] \leftarrow 0 \\
\quad \quad S2[i] \leftarrow 0 \\
\quad \quad S1[1] \leftarrow 1 \\
\quad \quad L \leftarrow 1 \\
\quad \text{for } i \leftarrow 2 \text{ to } n \\
\quad \quad \text{if } A[i] > A[S1[L]] \\
\quad \quad \quad S1[L] \leftarrow S1[L] \\
\quad \quad \quad S2[L] \leftarrow i
\]

2
\[ L \leftarrow L + 1 \]
\[ S1[L] \leftarrow i \]
\[ S2[i] \leftarrow S1[L - 1] \]
\textbf{else if} \ A[i] < A[S1[1]]
\[ S1[1] \leftarrow i \]
\textbf{else for} \ j \leftarrow L - 1 \textbf{downto} 1
\textbf{if} \ A[S1[j]] < A[i] < A[S1[j + 1]]
\[ S1[j + 1] \leftarrow i \]
\[ S2[i] \leftarrow S1[j] \]
\textbf{break}
\[ p \leftarrow S1[L] \]
\textbf{for} \ i \leftarrow L \textbf{downto} 1
\[ LMIS[i] \leftarrow A[p] \]
\[ p \leftarrow S2[p] \]

Notice that \( A[S1[i]] \leq A[S1[j]] \), where \( i < j \). We can improve the above algorithm from \( O(n^2) \) to \( O(n \log n) \) by using binary search in appending the current number.