1. (a) Arrange the objects in the nonincreasing order of $p_i/w_i$'s: $(1, 7, 5, 3, 6, 2, 4)$. Optimal solution $= 12 + 6 + 8 + 5 + 12 + 24/5 = 47.8$

(b) Consider two objects: $p_1 = 2, w_1 = w$ and $p_2 = 1, w_2 = 1$ and $m = 1$. Let $w > 2$. $F^*(I) = 1$ and $\tilde{F}(I) = 2/w$. As $w \to \infty$, $F^*(I)/\tilde{F}(I) \to \infty$.

(c) Consider the objects: $p_1 = 1, w_1 = 1$ and $p_2 = p, w_2 = 2$ and $m = 1$. Let $p > 2$. $F^*(I) = p/2$ and $\tilde{F}(I) = 1$. As $p \to \infty$, $F^*(I)/\tilde{F}(I) \to \infty$.

2. Let $C$ be a cycle with $k$ vertices. Let the edge $e$ of $C$ with the maximum weight be a part of a minimum-cost spanning tree of $G$. There exists at least one edge, $e'$, of $C$ which is not a part of the minimum-cost spanning tree (k vertices can be connected by $k - 1$ edges and $C$ contains $k$ edges). By replacing $e$ in the spanning tree with $e'$, we can obtain a new spanning tree whose cost will be less than that of the original minimum-cost spanning tree. This is a contradiction and hence, the edge with the maximum weight of $C$ can not be a part of a minimum-cost spanning tree of $G$.

3. In the graph $G$ replace every edge with a path of length equal to the weight on the edge. For example, if the weight on the edge $e$ is $c$ (where $c$ is an integer), replace this edge with a path of length $c$ (where each edge has unit cost). Now perform a BFS starting from the source to figure out the shortest path from the source to every other node. The algorithm takes $O(|V| + |E|)$ time.

4. Let $X = x_1, x_2, x_3, \ldots, x_n$ and $Y = y_1, y_2, y_3, \ldots, y_m$.

Let $L(i, j)$ represent the length of the longest common subsequence between $x_1, x_2, \ldots, x_i$ and $y_1, y_2, \ldots, y_j$ such that the common subsequence ends at $x_i$ and $y_j$ in $X$ and $Y$ respectively. Assuming that $L[i-1, j-1]$ has already been calculated, compute $L[i, j]$ as follows:

\[ \text{if } x[i] = y[j] \text{ then } L[i, j] = L[i-1, j-1] + 1 \]
\[ \text{else } L[i, j] = 0. \]

Compute $L[i, j]$ for $0 \leq i \leq n; 0 \leq j \leq m$. Scan through all these values and pick the largest $L[i, j]$. Computing $L[i, j]$ from $L[i-1, j-1]$ takes constant time. Thus the total run time is $O(nm)$.

5. (a) $C(i, j) = \min_{i \leq k \leq j} \{C(i, k) + C(k + 1, j) + D(i - 1)D(k)D(j)\}$

(b) for $i = 1$ to $r$ do

\[ \text{for } j = i + 1 \text{ to } r \text{ do } \]
\[ \text{for } k = i \text{ to } j \text{ do } \]

\[ C[i, j] = \min(C[i, j], C[i, k] + C[k + 1, j] + D(i - 1)D(k)D(j)) \]
6. The coefficients of the polynomial are given by \( \binom{n}{i} \), \( i = 0, 1, \ldots, n \).

Since \( \binom{n}{i} = \binom{n}{n-i} (n-i+1)/i \), the coefficients can be computed in time \( O(n) \).

FFT can be used to multiply two \( n \)th degree polynomials in \( O(n \log n) \) time. We can compute the coefficients of \((1+x)^n\) by multiplying \((1+x)^{n/2}\) and \((1+x)^{n/2}\). If \( T(n) \) is the time needed to compute \((1+x)^n\), then, \( T(n) = 2T(n/2) + O(n \log n) \), which solves to \( O(n \log^2 n) \).

7. Let \( A \) be a Toeplitz matrix and \( B \) be an \( n \times 1 \) vector. Let’s consider the multiplication of the lower triangular part of \( A \) (including the main diagonal elements) with \( B \).

Let the elements of \( A \) be the following:

\[
\begin{align*}
a_{n,n} &= a_{n-1,n-1} = a_{n-2,n-2} = \cdots = a_{2,2} = a_{1,1} = a_1 \\
a_{n,n-1} &= a_{n-1,n-2} = a_{n-2,n-3} = \cdots = a_{2,1} = a_2 \\
a_{n,n-2} &= a_{n-1,n-3} = a_{n-2,n-4} = \cdots = a_{3,1} = a_3 \\
\vdots \\
a_{n,1} &= a_n
\end{align*}
\]

Let the elements of \( B \) be the following:

\[
\begin{align*}
b_{1,1} &= b_1, b_{2,1} = b_2, \ldots, b_{n,1} = b_n
\end{align*}
\]

Multiplication of the lower triangular part of \( A \) with \( B \) gives the following:

\[
\begin{bmatrix}
a_1b_1 \\
a_2b_1 + a_1b_2 \\
a_3b_1 + a_2b_2 + a_1b_3 \\
\vdots
\end{bmatrix}
\]

We can notice that the above is nothing but the multiplication of two polynomials \((a_1 + a_2x + a_3x^2 + \ldots)\) and \((b_1 + b_2x + b_3x^2 + \ldots)\).

Since the polynomials can be multiplied in \( O(n \log n) \) time, the matrices can also be multiplied in \( O(n \log n) \) time. Multiplication of the upper triangular elements of \( A \) with \( B \) is symmetrical to the above and it would not affect the asymptotic complexity.

8. Using Taylor series expansion for \( f(\cdot) \),

\[
f(a + x) = f(a) + xf'(a) + \frac{x^2}{2!}f''(a) + \ldots + \frac{x^n}{n!}f^n(a)
\]

where \( f^i(x) \) stands for the \( i \)th derivative of \( f(x) \). Let \( F(x) \) denote \( f(a + x) \). Evaluate \( F(x) \) at the \( n \)th roots of unity. This can be done in \( O(n \log^2 n) \) time as was mentioned in class (see Section 9.5 in the text – An \( n \)th degree polynomial can be evaluated at \( n \) arbitrary points in \( O(n \log^2 n) \) time). Then, use inverse FFT to compute the coefficients of \( F(x) \). This can be done in \( O(n \log n) \) time. Once the coefficients of \( F(x) \) are known, it is easy to determine the derivatives.

Run time = \( O(n \log^2 n) + O(n \log n) + O(n) = O(n \log^2 n) \).

**Another solution:** Let \( f(x) = a_nx^n + a_{n-1}x^{n-1} + \cdots + a_1x + a_0 \). Then, \( f'(x) = na_nx^{n-1} + (n-1)a_{n-1}x^{n-2} + \cdots + 2a_2x + a_1; f''(x) = n(n-1)a_nx^{n-2} + (n-1)(n-2)a_{n-1}x^{n-3} + \cdots + 2a_2; f'''(x) = n(n-1)(n-2)a_nx^{n-3} + (n-1)(n-2)(n-3)a_{n-1}x^{n-4} + \cdots + 6a_3; \) and so on. Now consider the following two polynomials: \( A(x) = (n!a_n)x^{n-1} + ((n-1)!a_{n-1})x^{n-2} + \cdots + \)
Compute the product of $A(x)$ and $B(x)$ in $O(n \log n)$ time. We can obtain the required derivatives from the coefficients of this product. The total run time is $O(n \log n)$. However note that the coefficients of $A(x)$ could become very large creating practical difficulties.