Problem 1 (20 points, Problem 15.1-2)

Solution. Here is a counterexample to prove that greedy algorithm doesn’t provide an optimal solution every time: Let $p_1 = 1, p_2 = 20, p_3 = 33, p_4 = 36$. Let the length of the rod be 4 inches. As we run the Greedy algorithm, the first cut for the rod would be of length 3 whose density is maximum. As a result, the total price would be 34. However, if we cut the rod into two pieces of length 2, the total price would be 40 which is optimal. Hence, the greedy algorithm does not always produce an optimal solution.

Problem 2 (30 points, Problem 15.1-3)

Solution. Let $r_n$ be the max revenue for length $n$. Then we have $r_n = \max_{1 \leq i < n} \{p_n, p_i + r_{n-i} - c\}$.

Pseudocode:

<table>
<thead>
<tr>
<th>Input: $p, n, c$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output: $r_n$</td>
</tr>
<tr>
<td>1: let $r_0 = 0$;</td>
</tr>
<tr>
<td>2: for $j = 1$ to $n$ do</td>
</tr>
<tr>
<td>3: let $q = p_j$</td>
</tr>
<tr>
<td>4: for $i = 1$ to $j - 1$ do</td>
</tr>
<tr>
<td>5: let $q = \max{q, p_i + r_{j-i} - c}$</td>
</tr>
<tr>
<td>6: end for</td>
</tr>
<tr>
<td>7: let $r_j = q$</td>
</tr>
<tr>
<td>8: end for</td>
</tr>
</tbody>
</table>

The time complexity of this algorithm is $O(n^2)$. 
Problem 3 (50 points, 25 points each, Problem 15.7)

Solution. a. We define a $k \times n$ matrix $m$ as follows:

$$m[i, j] = \begin{cases} 1 & \text{if there is a path from } v_0 \text{ to } v_j \text{ with a sequence of sounds } < \sigma_1, ..., \sigma_i > \text{ as its label} \\ 0 & \text{otherwise} \end{cases}$$

The matrix can be calculated as follows:

$$m[i, j] = \begin{cases} 1 & \text{if there is an edge } (v_h, v_j) \text{ s.t. } \sigma(v_h, v_j) = \sigma_i \text{ and } m[i - 1, h] = 1 \\ 0 & \text{otherwise} \end{cases}$$

Pseudocode:

The running time is $O(kn^2)$.

b. Let $f[i, j]$ denote the maximum probability of a path from $v_0$ to $v_j$ with $< \sigma_1, ..., \sigma_i >$ as its label.

$$f[i, j] = \max_h \{ f[i - 1, h] p(v_h, v_j) \}$$

The pseudocode is similar to (a). The time complexity is $O(kn^2)$

---

Algorithm 1 Pseudo-code for Problem 2(a)

**Input:** $G = (V, E), s$

**Output:** a path with label $s$ or "NO-SUCH-PATH"

1. let $m[i, j] = 0 \ (i = 1, ..., k; \ j = 0, ..., n - 1)$
2. for $j = 0$ to $n - 1$ do
3.   if $\sigma(v_0, v_j) == s_1$ then
4.     let $m[1, j] = 1$
5.   end if
6. end for
7. for $i = 2$ to $k$ do
8.   for $j = 0$ to $n - 1$ do
9.     if there is an edge $(v_h, v_j)$ s.t. $\sigma(v_h, v_j) = s_i$ and $m[i - 1, h] == 1$ then
10.       let $m[i, j] = 1$
11.     end if
12.   end for
13. end for
14. let path = ""
15. for $j = 0$ to $n - 1$ do
if \( m[k, j] == 1 \) then

\[ \text{path} = \text{path} + v_j \]

let \( v_b = v_j \)

for \( i = k \) downto 2 do

if there is an edge \((v_a, v_b)\) s.t. \( \sigma(v_a, v_b) = \sigma_i \) and \( m[i-1, a] == 1 \) then

let \( v_b = v_a \)

let \( \text{path} = \text{path} + v_a \)

end if

end for

let \( \text{path} = \text{path} + v_0 \)

return \text{path}

end if

end for

return "NO-SUCH-PATH"

---

**Algorithm 2** Pseudo-code for Problem 2(b)

**Input:** \( G = (V, E), s, p \)

**Output:** a path with label \( s \) or "NO-SUCH-PATH"

1. let \( f[i, j] = 0 \) (\( i = 1, \ldots, k; j = 0, \ldots, n - 1 \))
2. for \( j = 0 \) to \( n - 1 \) do
3. if \( \sigma(v_0, v_j) == s_1 \) then
4. let \( f[1, j] = p(v_0, v_j) \)
5. end if
6. end for
7. for \( i = 2 \) to \( k \) do
8. for \( j = 0 \) to \( n - 1 \) do
9. \( f[i, j] = \max_{h, s.t. \sigma(v_h, v_j) = \sigma_i} \{ f[i - 1, h]p[v_h, v_j] \} \)
10. end for
11. end for
12. let \( \text{path} = "" \)
13. let \( i = -1, pr = 0 \)
14: for \( j = 0 \) to \( n - 1 \) do \\
15: \quad \text{if } f[k, j] > pr \text{ then} \\
16: \quad \text{let } pr = f[k, j] \\
17: \quad \text{let } i = j \\
18: \quad \text{end if} \\
19: \text{end for} \\
20: \text{if } i == -1 \text{ then} \\
21: \quad \text{return } "\text{NO-SUCH-PATH}" \\
22: \text{end if} \\
23: \text{let } path = path + v_i \\
24: \text{for } j = k \text{ down to } 2 \text{ do} \\
25: \quad \text{find } h \text{ s.t. } f[j - 1, h]p(v_h, v_i) = f[j, i] \text{ and } \sigma(v_h, v_i) = \sigma_j \\
26: \quad \text{let } v_i = v_h \\
27: \quad \text{let } path = path + v_h \\
28: \text{end for} \\
29: \text{let } path = path + v_0 \\
30: \text{return path}