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:

Input: p, n, c
Output: r_n

let $r_0 = 0$;
for $j = 1$ to n **do**
    let $q = p_j$
    for $i = 1$ to $j - 1$ **do**
        let $q = \max\{q, p_i + r_{j-i} - c\}$
    end for
    let $r_j = q$
end for

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.

$f[i,j] = \max_{h \ s.t. \sigma(v_h, v_j) = \sigma_i} \{f[i-1,h]p(v_h, v_j)\}$

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

Algorithm 1 Psedo-code for Problem 2(a)

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

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

let $m[i,j] = 0$ $(i = 1, ..., k; j = 0, ..., n-1)$
for $j = 0$ to $n-1$ **do**
  if $\sigma(v_0, v_j) == s_1$ **then**
    let $m[1,j] = 1$
  end if
end for
for $i = 2$ to k **do**
  for $j = 0$ to $n-1$ **do**
    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**
      let $m[i,j] = 1$
    end if
  end for
end for
let $path$ = ""
for $j = 0$ to $n-1$ **do**

```
    if m[k,j] == 1 then
      path = 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. σ(v_a, v_b) = σ_i and m[i-1, a] == 1
        then
          let v_b = v_a
          let path = path + v_a
        end if
      end for
      let path = path + v_0
      return path
    end if
end for
return "NO-SUCH-PATH"
```

Algorithm 2 Psedo-code for Problem 2(b)

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

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

```
let f[i,j] = 0 (i = 1, ..., k; j = 0, ..., n-1)
for j = 0 to n-1 do
  if σ(v_0, v_j) == s_1 then
    let f[1,j] = p(v_0, v_j)
  end if
end for
for i = 2 to k do
  for j = 0 to n-1 do
    f[i,j] = max_{h s.t. σ(v_h, v_j) = σ_i} {f[i-1, h] p[v_h, v_j]}
  end for
end for
let path = ""
let i = -1, pr = 0
```

for $j = 0$ to $n-1$ **do**
    if $f[k,j] > pr$ **then**
        let $pr = f[k,j]$
        let $i = j$
    end if
end for
if $i == -1$ **then**
    return "NO-SUCH-PATH"
end if
let $path = path + v_i$
for $j = k$ down to 2 **do**
    find h s.t. $f[j-1,h]p(v_h, v_i) == f[j,i]$ and $\sigma(v_h, v_i) == \sigma_j$
    let $v_i = v_h$
    let $path = path + v_h$
end for
let $path = path + v_0$
return path