Exercise 1

1. *What is your top-choice for this problem? (1)*

   The top-choice is the position of the last space.

2. What is your subproblem for the top-choice you described above? (1)

   Let $y = y_1 y_2 \dots y_n$ be a string that we want to decode. A subproblem of this would be to decode $y_1 y_2 \dots y_m$ for any $1 \leq m \leq n$.

3. Write the subproblem for your top-choice as a recurrence relation. (2)

   If maxQuality$(y_1 y_2 \dots y_n)$ is the highest quality we can get from the sequence of letters $y_r y_2 \dots y_n$ we see that

$$\text{maxQuality}(y_1 y_2 \dots y_n)$$

$$= \max\left(q(y_1 y_2 \dots y_m), \max_{2 \leq m \leq n} \left(q(y_m y_{m+1} \dots y_n) + \text{maxQuality}(y_1 y_2 \dots y_{m-1})\right)\right)$$

4. Prove the optimality principle for your subproblem for the corresponding top-choice.(3)

   Consider a sequence of indices $S = \{i_1, \dots, i_k\}$ that gives us the choice of spaces that has the highest quality. Say $i_k = m$. Then $S \backslash \{i_k\} = \{i_1, \dots, i_{k-1}\}$ is a solution for the subproblem with $y_1 \dots y_{m-1}$. Let $T = \{j_1, \dots, j_{k'}\}$ be an optimal solution of the subproblem $y' = y_1 y_2 \dots y_{m-1}$. We know that

$$q(y_1 y_2 \dots y_{j_1 - 1}) + q(y_{j_1} y_{j_1+1} \dots y_{j_2 - 1}) + \dots + q(y_{j_{k'}} y_{j_{k'}+1} \dots y_{i_k - 1}) \geq \tag{1}$$

$$q(y_1 y_2 \dots y_{i_1 - 1}) + q(y_{i_1} y_{i_1+1} \dots y_{i_2 - 1}) + \dots + q(y_{i_{k-1}} y_{j_{k-1}+1} \dots y_{i_k - 1}) \tag{2}$$

   Because $S \backslash \{i_k\}$ is a solution. Note that $S' = \{j_1, \dots, j_{k'}, i_k\}$ is also a possible splitting of $y_1 \dots y_n$. Because of (1) and (2) we see that

$$q(y_1 y_2 \dots y_{j_1 - 1}) + \dots + q(y_{j_{k'}} y_{j_{k'}+1} \dots y_{i_k - 1}) + q(y_{j_k} y_{j_k+1} \dots y_n) \geq \tag{3}$$

$$q(y_1 y_2 \dots y_{i_1 - 1}) + \dots + q(y_{i_{k-1}} y_{j_{k-1}+1} \dots y_{i_k - 1}) + q(y_{i_k} y_{j_k+1} \dots y_{i_n}) \tag{4}$$

   Thus S' has a higher or equal quality than S and therefore it is again an optimal solution.

5. Write a pseudocode for your algorithm and analyze its runtime. (3)

   If $y_1 y_2 \dots y_n$ is the string the following algorithm will give us the optimal the space indices.

```
1       //O(n^2)
2       for(m=1,...,n){
3           //O(1)
4           max[m] = q(y_1...y_m)
5           //O(1)
6           amount[m] =0
7           //O(n)
8           //If m=1 it will loop zero times
9           for(l=2,...,m){
10              //O(1)
11              if(max[m] < q(y_l...y_m) + max[l-1]){
12                  //O(1)
13                  max[m] = q(y_l...y_m) + max[l-1]
14                  //O(1)
15                  amount[m] = 1+amount[l-1]
16              }
17          }
18      }
19      //O(1)
20      k = amount[n]
21      //O(1)
22      previous_index = n
23      //O(n)
24      for(m=n,...,1){
25          //O(1)
26          if(max[m] + q(y_{m+1}...y_previous_index) == max[previous_index]){
27              //O(1)
28              i_k = m
29              //O(1)
30              k= k-1
31              //O(1)
32              previous_index = m
33      }
34      return(i_1...i_k)
```

We shall now analyze the runtime. Because q takes O(1) time it should be clear that lines 9-15 take O(1) time. Because the for-loop in line 8 repeats $m-1 \leq n$ time we see that 8-16 takes O(n) time. We see that 3-7 take constant time, so because $O(1)+O(n) = O(n)$ we see that 3-16 take $O(n)$ time. We see that the for loop in line 2 repeats n times, so 1-17 take $O(n^2)$ time.

We see that 24-31 takes constant time and is repeated n times, so 23-32 is O(n) time. Because $O(n)+O(n^2) = O(n^2)$ we see that the time complexity is $O(n^2)$. This is a lot better than going through all possibilities which had a time complexity of $O(2^n)$