1. Provide an example when the Boyer-Moore bad character rule will result in a running time of \(O(m \times n)\) (\(m\) - length of text, \(n\) - length of pattern).

For extra credit (+2 points) provide an example when the Boyer-Moore algorithm would perform fewer comparisons when the bad character rule is used alone, instead of combining it with the good suffix rule (problem 7 in Chapter 2 of Gusfield).

2. Problem 9 in Chapter 2 of Gusfield:
   Let \(l'(i)\) denote the length of the largest suffix of \(P[i..n]\) that is also a prefix of \(P\), if one exists, otherwise let \(l'(i)\) be 0.

   **Theorem:** \(l'(i)\) equals the largest \(j \leq |P[i..n]|\) (i.e. \(j \leq n - i + 1\)) s.t. \(N_j = j\).

   Prove the theorem and describe an algorithm that computes the \(l'(i)\) values in linear time. Explain the correctness of the algorithm.

   **Hint:** algorithm is similar to the accumulation of the \(L'(i)\) values in the execution of Boyer-Moore.

3. Problem 6 in Chapter 3 of Gusfield:
   For each of the \(n\) prefixes of \(P\), we want to know whether prefix \(P[1..i]\) is a periodic string. That is, for each \(i\) we want to know the largest \(k > 1\) (if there is one) s.t. \(P[1..i]\) can be written as \(a^k\) for some string \(a\). Of course, we also want to know the period. Show how to determine this for all \(n\) prefixes in linear time in the length of \(P\).

   **Hint:** Z-algorithm.