Problem 1: Let \( G(V, E) \) be a complete undirected graph (i.e., there is an edge between each pair of vertices in \( V \)) such that for each edge \( \{u, v\} \), there is a positive distance value \( d(u, v) \). The **diameter** of \( G \) is the largest value among all the edge distances in \( G \). (If \( G \) contains only one node, the diameter of \( G \) is zero.) When we partition \( V \) into two subsets \( V_1 \) and \( V_2 \), we get two complete subgraphs \( G_1(V_1, E_1) \) and \( G_2(V_2, E_2) \), where \( E_1 \) and \( E_2 \) are respectively the set of edges in the complete graph induced on \( V_1 \) and \( V_2 \) respectively. Thus, the diameters of \( G_1 \) and \( G_2 \) are well defined. Consider the following problem.

**Given:** A complete undirected graph \( G(V, E) \) with a positive distance value \( d(u, v) \) for each edge \( \{u, v\} \) and positive numbers \( \alpha_1 \) and \( \alpha_2 \), where \( \alpha_1 \geq \alpha_2 \).

**Question:** Is there a partition of the node set \( V \) into two subsets \( V_1 \) and \( V_2 \) such that the diameter of the subgraph \( G_1(V_1, E_1) \) is \( \leq \alpha_1 \) and the diameter of the subgraph \( G_2(V_2, E_2) \) is \( \leq \alpha_2 \)?

Give a polynomial time algorithm for the above problem. Your answer must provide a high level description of the algorithm, a proof of its correctness and a proof that its running time is a polynomial function of \( |V| \).

**Hint:** Use a reduction to 2SAT (i.e., the Satisfiability problem in which each clause has at most two literals). In analyzing the running time of your algorithm, you may assume that any 2SAT instance with \( p \) variables and \( q \) clauses can be solved in \( O(p + q) \) time.

Problem 2: All the strings considered in this problem are bit strings (i.e., they are composed of 0’s and 1’s). Given a text string \( S \) and a set \( P = \{p_1, p_2, \ldots, p_m\} \) of \( m \) pattern strings, we say that \( S \) is **realizable using** \( P \) if \( S \) can be split into one or more substrings such that each substring is a member of \( P \). (A string in \( P \) may be used any number of times in such a realization.) This problem asks you to devise a dynamic programming algorithm to realize \( S \) from \( P \) using a *minimum* number of pattern substrings, if possible. Here is an example to illustrate the problem.

**Example:** Suppose we have a collection of four pattern strings \( p_1, p_2, p_3 \) and \( p_4 \), where \( p_1 = 1, p_2 = 01, p_3 = 101 \) and \( p_4 = 111 \). Consider the text string \( S = 101111101 \). Two ways of realizing the text string \( S \) using the pattern strings in \( P \) are as follows. (We use the symbol ‘-’ between substrings just to show how the string \( S \) is split.)

(a) 1-01-111-1-01 : Number of substrings = 5
(b) 101-111-101 : Number of substrings = 3
In this example, it can be seen that the minimum number of substrings needed to realize $S$ is three. On the other hand, one can verify that the string 00011 can’t be realized using $P$. Thus, the general problem can be formulated as follows.

**Given:** A set $P$ of $m$ pattern strings $p_1, p_2, \ldots, p_m$, each of length at most $k$ and a text string $S = s_1s_2 \ldots s_n$ of length $n$.

**Requirement:** Compute the minimum number of substrings needed to realize $S$. (If $S$ is not realizable, we use the convention that the minimum number of substrings needed is $\infty$.)

Give a dynamic programming algorithm for the above problem. You need to compute only the minimum number of substrings needed to realize $S$. The running time of your algorithm must be a polynomial in $n$, $m$ and $k$.

You must clearly specify what information is stored in the table which is maintained as part of your algorithm. You must also specify equation(s) showing how the entries in the table are computed and how you arrived at the equation(s).

Recall that each pattern string in $P$ is of length at most $k$. Given a string $X$ with known length $\ell$ and a pattern string $p_j$ from $P$, you can assume that determining whether $X$ and $p_j$ are identical can be done in $O(k)$ time.

Be sure to include (high level) pseudocode for your algorithm and explain why the running time of your algorithm is a polynomial in $n$, $m$ and $k$.

**Note:** Although there is no need to compute a partition of $S$ into a minimum number of substrings, you are strongly encouraged to think about how to obtain such a realization.

**Problem 3:** Suppose $A = a_1a_2a_3\ldots a_n$ and $B = b_1b_2b_3\ldots b_m$ are strings with $m \leq n$. We say that $B$ is a subsequence of $A$ if there exist indices $1 \leq i_1 < i_2 < \ldots < i_m \leq n$ such that for all $j, 1 \leq j \leq m$, $b_j = a_{i_j}$.

**Example:** Suppose $A$ is the string axbyyczdz and $B$ is the string xyz, then $B$ is a subsequence of $A$. However, for the same $A$, if $C$ is the string xxyz, then $C$ is not a subsequence of $A$.

For any string $B$ and for each $r \geq 1$, define $B^r$ to be the string obtained from $B$ by repeating each character of $B$ exactly $r$ times. For example, if $B$ is the string xxyz, then $B^3$ is the string xxyyxxxxzzz.

Given strings $A$ and $B$, give an algorithm to determine the largest integer $r$ such that $B^r$ is a subsequence of $A$. The running time of your algorithm should be $O(n \log(n/m))$.

Your answer must include a high level description of the algorithm, a proof of its correctness and a proof that its running time is $O(n \log(n/m))$.

**Hint:** You may find it useful to first design an algorithm with a running time of $O(p)$ for the following problem: Given two strings $P$ and $Q$ whose lengths are $p$ and $q$ respectively, where $q \leq p$, determine whether $Q$ is a subsequence of $P$. 
