**Problem 1: Palindromes**

Let \( S = (s_1 s_2 \cdots s_n) \) be the input sequence. Define \( A[i, j] \) as the number of ways one can remove a few symbols (maybe 0) from sequence \((s_i \cdots s_j)\) such that the rest of sequence \((s_i \cdots s_j)\) becomes a palindrome. Write a dynamic program to compute \( A[i, j] \) based on \( A[i + 1, j] \), \( A[i, j - 1] \), and \( A[i + 1, j - 1] \).

**Problem 2: Tree**

For each vertex \( v \), compute the final answer to query “find \( v \)” and store it in \( result[v] \). You can fill array \( result \) by one DFS.

- For each query “find \( v \)”, you can output \( result[v] \).
- For each query “change \( v \) \( w \)”, you should update \( result[u] \) for each vertex \( u \) in the subtree rooted at \( v \) (including \( v \)). Design an algorithm for updating array \( result \) by one DFS from vertex \( v \). Note that there at most 100 queries of this type.

**Problem 3: Increasing Shortest Path**

Sort all edges regarding their weights and let \((e_1, e_2, \cdots, e_m)\) be the result, i.e., \( w_{e_1} \leq w_{e_2} \leq \cdots \leq w_{e_m} \).

Now, consider a query which is represented by \( A \) and \( B \). This means you need to find the shortest path (the path with minimum sum of weights of its edges) which goes from node \( A \) to node \( B \) such that the weights of the edges in that path are in increasing order along the path.

Let \( B[i] \) be the minimum sum of weights for a path starting at node \( A \) and ending at edge \( e_i \) which satisfies the given constraints. Write a dynamic program for computing \( B[1], B[2], \cdots, B[m] \).

Given array \( B \), what would be the final answer?