To create the C++ program for the "Route" problem, we need to implement the route function as specified in the Day1\_Route.pdf file. The goal is to define routes for

M cars (edges) with given costs W, such that the "economical allocation" (which is the Minimum Spanning Tree, or MST, of the graph formed by these routes) has the highest possible total cost.

To maximize the MST cost, we should assign the N-1 most expensive available car costs to edges that form a spanning tree, and then assign the remaining cheaper costs to other valid edges. A simple path graph (connecting nodes 1-2, 2-3, ..., N-1-N) always forms a spanning tree.

Here's the C++ implementation strategy:

1. **Function Signature:** Implement std::vector<std::pair<int, int>> route (int N, std::vector<int> W).
2. **Identify Most Expensive Cars:** The W vector contains the costs of M cars, sorted from least to greatest. Thus, the

N-1 most expensive cars correspond to the last N-1 elements in W (i.e., W[M - (N-1)] through W[M-1]).

1. **Assign Path Edges:** Assign these N-1 most expensive cars to form a simple path connecting N nodes (e.g., node 1 to node 2, node 2 to node 3, ..., node N-1 to node N). This ensures a spanning tree is formed using the highest possible costs.
2. **Assign Remaining Edges:** The remaining M - (N-1) cars (which have smaller costs, W[0] through W[M-(N-1)-1]) need to be assigned to other valid, distinct edges. We can systematically generate valid (u,v) pairs, avoiding self-loops (u == v) and duplicates. A std::set is used to keep track of assigned edges (normalized as min(u,v), max(u,v)) to ensure uniqueness.
3. **No main function or I/O:** As per the problem's specific requirements for competitive programming, the solution will only contain the route function and no main function, nor any input/output operations like cin or cout.