Yes, this algorithm will find a minimum spanning tree.

Proof of Correctness:
First we are going to prove that if every edge has a distinct cost, there will be only one MST. We use contradiction: Assume T_1 and T_2 are distinct MSTs. Let edge AB be in T_1 but not T_2. Then T_2 have to contain a route from A to B other than AB. If we add AB to T_2, we get a cycle. If the other edges in this "cycle" are in T_1, then T_1 would contain a cycle since AB is in T_1, so the "cycle" must contain at least one edge e that is not in T_1. So the cost of e must be greater than AB. If we replace e by AB the total cost would be smaller, which is a contradiction.

Now, for this algorithm, starting with every vertex, adding the lowest cost edge belonging to its own tree. Basically it is "parallel" implementing Prim's algorithm on each vertex. For Prim's algorithm, we have proved that its correctness in class and it can generate a MST without deleting any edges during the process. Hence, in the process, every edge selected is a part of the MST (and the only MST as we proved in the last paragraph). For each pair of the vertex, If the edge link of them is the lowest cost of both of them, these two vertices can be merged together into a single vertex. Hence, in the end there will be a tree where each vertex corresponds to a component connected including all the edges chosen, and the edges belong to the only MST, which proves the correctness of this algorithm.

Complexity Analysis:
For each step, the number of small trees will be reduced by at least a factor of 2. So for the worst case, we assume it is reduced by the factor 2 every time, so there will be $O(\log n)$ steps. During each step, the algorithm can be implemented in $O(m)$. Hence, the number of total iterations in the worst case will be $O(m \log n)$.