设 $G = (V, E)$ 是一个连通图，T 是 G 的任意一棵最小生成树。我们需要构造一个边的“有效排序”，使得 Kruskal 算法在应用该排序时能输出 T。

1. **构造特殊的边排序策略**：我们对图 G 中所有的边 E 进行排序，遵循以下规则：
   (a) 首先，按照所有边（无论是否在 T 中）的**费用** c_e **递增**的顺序进行排序。
   (b) **对于费用相同的边**：
      - 如果边 e **属于** T (即 $e \in T$)，则将其排在所有与它费用相同但**不属于** T 的边之前。
      - 在满足上述条件的情况下，费用相同且都属于 T 的边，以及费用相同且都不属于 T 的边，它们各自内部的相对顺序可以任意。

   这个排序是一个“有效排序”，因为它确保了边的费用序列是非降的。

2. **运行 Kruskal 算法**：使用这个特殊构造的边排序，运行 Kruskal 算法。Kruskal 算法会依次考虑排序后的每条边，如果添加这条边不会形成环，则将其加入到当前正在构建的森林中。

3. **证明输出结果为** T：我们来证明 Kruskal 算法将最终构建出树 T。

   - 考虑 Kruskal 算法在遍历排序后的边列表时。当它遇到任何一条边 e 时，如果 $e \in T$，由于我们的排序策略，它将优先于所有与它费用相同但不在 T 中的边被考虑。
   - **关键性质**：最小生成树 T 中的任何一条边都不会与 T 中已有的其他边形成环。这是因为如果 T 中的某条边 (u, v) 与 T 中其余边形成环，那么移除 (u, v) 并用环中另一条连接 u, v 的边 (x, y) 替换，如果 $c_{x,y} \leq c_{u,v}$，则可以得到一个总费用更小或相等的生成树，这与 T 是最小生成树的假设矛盾（除非 T 不唯一）。但更普遍地，Kruskal 算法的正确性依赖于它只添加不会形成环的边。
   - 因此，当 Kruskal 算法处理到 T 中的边时，它总能将其加入到正在构建的森林中，因为这些边本身不会形成环，且它们在同费用边中具有优先权。
   - Kruskal 算法会持续添加边，直到形成一个包含 $n - 1$ 条边的连通无环图，即一棵生成树。由于我们优先选择了 T 中的边，并且 T 本身就是一棵包含 $n - 1$ 条边的生成树，Kruskal 算法在选择了所有 T 中的边后，将停止并输出 T。即使存在其他与 T 中边费用相同的边，因为 T 已经构成一棵树，再添加其他边会形成环，Kruskal 算法不会选择它们。

综上所述，通过构造一个适当的有效边排序，我们可以确保 Kruskal 算法的输出恰好是任意给定的最小生成树 T。这表明 Kruskal 算法可以“找到”图 G 的所有最小生成树（通过不同的有效排序）。