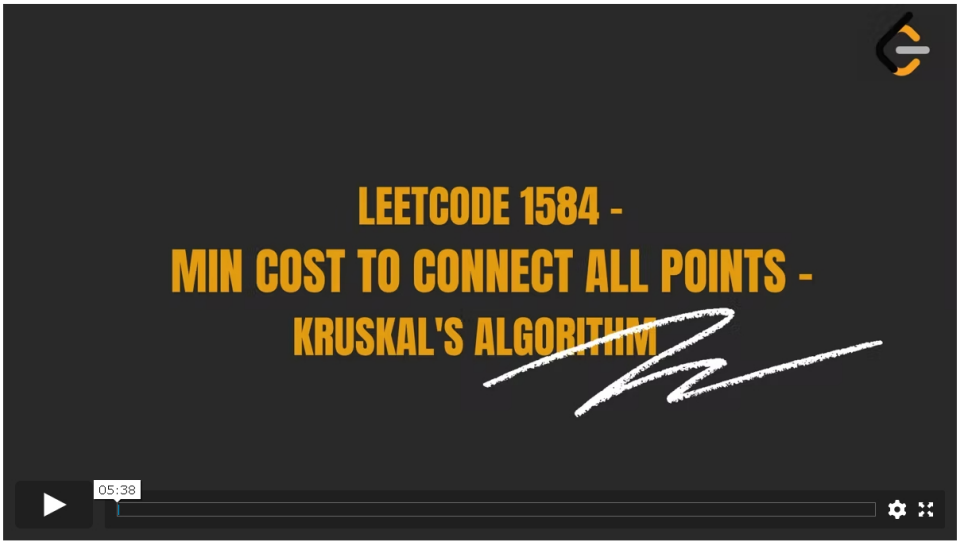
LeetCode 1584 - Min Cost to Connect All Points - Kruskal's Algorithm

Report Issue

Video Solution

In the following video, we'll approach the Min Cost to Connect All Points problem with Kruskal's Algorithm.



Implementation

C++ Java Python3 Copy Run Playground

```cpp
class Edge {
public:
    int point1;
    int point2;
    int cost;
    Edge(int point1, int point2, int cost)
        : point1(point1), point2(point2), cost(cost) {}
};

// Overload the < operator.
bool operator<(const Edge& edge1, const Edge& edge2) {
    return edge1.cost > edge2.cost;
}

class UnionFind {
public:
    UnionFind(int sz) : root(sz), rank(sz) {
        for (int i = 0; i < sz; i++) {
            root[i] = i;
            rank[i] = 1;
        }
    }

    int find(int x) {
        if (x == root[x]) {
            return x;
```

Complexity Analysis

- Time Complexity: $O(E \log E)$. Here, E represents the number of edges.
    - For Python, building a priority queue using heapify method takes $O(E)$ time, and we need $O(E \log E)$ time for popping out all the elements from the priority queue. In total, we need $O(E \log E)$ time in terms of Big-O notation.
    - For C++ and Java, building a priority queue takes $O(E \log E)$ time. Popping out all the elements from the queue takes $O(E \log E)$ time as well. Therefore, total time complexity for this solution is

$$O(E \log E) + O(E \log E) = O(E \log E).$$

- Space Complexity: $O(E)$. We need the space to store all the edges in a priority queue.