Approach

- Idea
  - if we view each start_i and end_i as nodes in a graph, elements in pairs as (directed) edges in the graph, then the problem is asking us to find a path in the corresponding directed graph.
  - the problem ask us to use up all edges, so we are basically asked to find a **Eulerian Path**, which is a path that walks through each edge **exactly once**.
    - if you are not familiar with the concept about **Eulerian Path**, there are some resources on [Wikipedia](), [CP-algo](), or other websites.
    - I will mention some important properties (without proof) of Eulerian Path below.
- Some Properties of Eulerian Path
  - I will use in[i] (and out[i]) to denote the in (and out) degree of a node i.
  - Existence:
    - A graph has an Eulerian Path if and only if
      1. we have out[i] == in[i] for each node i. Or
      2. we have out[i] == in[i] for all nodes i except **exactly two** nodes x and y, with out[x] = in[x] + 1, out[y] = in[y] - 1.
    - this problem guarantees that an Eulerian Path exists. So we don't need to check for its existence here.
    - In the first case (out[i] == in[i] for each node i), all Eulerian Paths are also **Eulerian Circuits** (Eulerian Path with starting point == ending point).
  - a node with out[i] == in[i] + 1 **must** be the starting point of an Eulerian Path (if there exists one).
- Algorithm

1. find the starting point of an Eulerian Path.
    - if we have out[i] == in[i] for all i, we can start at an arbitrary node.

2. perform **postorder DFS** on the graph, as we "walk" through an edge, we **erase** (or mark it visited) the walked edge.
    - we may reach the same node many times, but we have to pass each edge **exactly once**.
    - I use stack in my code for erasing edges.