- Given an array of paths where

paths = [[A, B], [B, C], ...]

- A travels to destination B, B travels to destination C
- It is guaranteed to have no loops, just a straight path.
- Find the destination.

Since we know there is only 1 final destination, we just search for a destination that is not a start.

Ex.) Input: [[A, B], [C, E], [E, A]]

end start

Our path is C → E → A → B

This means we could also find the start start w/ the same approach.

1.) Creat a set<string> S.

2.) Enumerate over all start cities inserting them into set.

3.) Enumerate over all end cities. The end city that does not exist in set is the final destination.

```
std::string final_destination(const std::vector<Route>& paths) {
        std::unordered_set<std::string> s;

        for (const auto& route : paths) {
            s.insert(route.start);
        }
        std::string result;
        for (const auto& route : paths)
            if (!s.contains(route.end)) {
                result = route.end;
                break;
            }
        }
        return result;
}
```