Main algorithm

```rust
/// Prints, and returns the powerset from a string input
pub fn into_powerset(set: String) -> Result<String, &'static str> {
    // println!("{:?}", set);
    let parsed_set: Vec<&str> = set.split(",").collect();

    let mut powerset = vec![String::from("")];

    let empty_set = parsed_set.iter().all(|val| val.trim().is_empty()); // empty
set/file is accepted

    // Handle empty set's'powerset
    if !(empty_set) {
        for set_el in parsed_set.iter() {
            for j in 0..powerset.len() {
                // Create new set based on current one adding the new item
                let new_set = if powerset[j].is_empty() {
                    format!("{}", set_el)
                } else {
                    format!("{},{}", &powerset[j], set_el)
                };
                // Concat new set to existing powerset
                powerset.push(
                    new_set
                        .trim_matches(char::is_whitespace) // trim end/start
whitspaces
                        .replace(char::is_whitespace, ""), // clean all other
whitspaces
                );
            }
        }
    }
    // Sort by string length (set size)
    powerset.sort_by(|a, b| compare_len(&a, &b)); // Sort the powerset
    let output = powerset.join("\n");
    print!("{}", output);
    Ok(output)
}
```

Time complexity

In the inner loop, we add the current element from the outer loop to all the existing subsets (the inner loop subsets length actually doubles each time). Given n elements in the original input set, The total number of subsets including the empty set is 2^n. So the time complexity is **O(n * 2^n)**. O(n) for the outer loop and O(2^n) for the inner loop

Space complexity

We have 2^n subsets and each subset has at most n elements. So the space complexity is **O(n * 2^n)**