Input: An array of unique integers
array = [1, 2, 3]

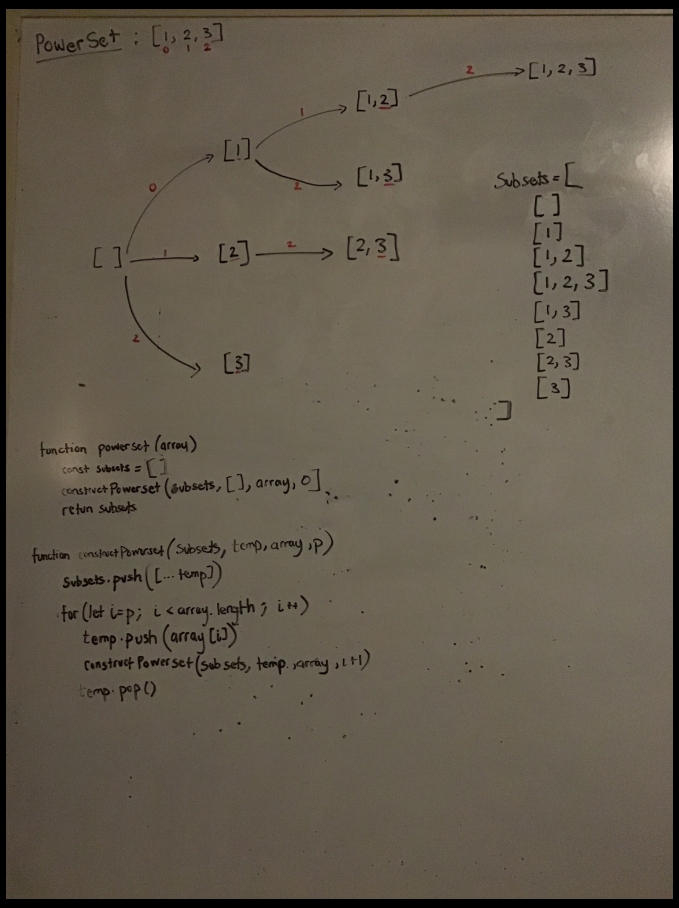
Output: returns its powerset
[[], [1], [2], [3], [1,2], [1,3], [2,3], [1,2,3]]

The powerset P(x) of a set X is the set of all subsets of X. For example, the powerset of [1,2] is:
[[], [1], [2], [1,2]]

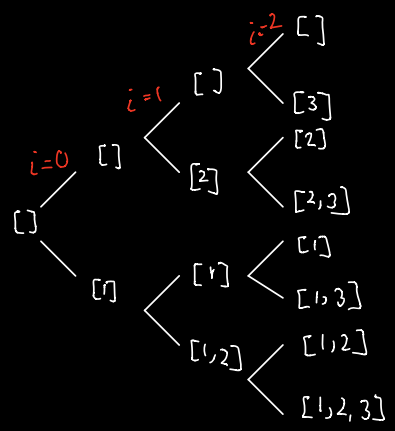
Note: Sets do not need to be in any particular order.

```
// O(n*2^n) time | O(n*2^n) space
function powerset(array) {
  const subsets = [];
  constructPowerset(subsets, [], array, 0);
  return subsets;
}

function constructPowerset(subsets, temp, array, p) {
  subsets.push([...temp]);
  for (let i = p; i < array.length; i++) {
    temp.push(array[i]);
    constructPowerset(subsets, temp, array, i + 1);
    temp.pop();
  }
}
```



Time: $O(n * 2^n)$ (where n is the # of elements in the input array. So we have 2^n ($n = 3$ here) "nodes" this is bc we have 2 choices at each node and at each node we copy up to n elements. Pushing and popping could be seen as $O(1)$ operations while copying is $O(n)$.



Space: $O(n * 2^n)$ since at each recursive call (2^n total) we are creating a temp array (of up to n elements at worst)