Daily Coding Problem #37

Problem

This problem was asked by Google.

The power set of a set is the set of all its subsets. Write a function that, given a set, generates its power set.

For example, given the set `{1, 2, 3}`, it should return `{{}, {1}, {2}, {3}, {1, 2}, {1, 3}, {2, 3}, {1, 2, 3}}`.

You may also use a list or array to represent a set.

Solution

To gain some intuition about this problem, let's try some examples:

- If we're given the empty set (`{}`), then the power set is a set with only the empty set in it: `{{}}`
- If we're given a set with one element in it (`{a}`), then the power set is a set with two sets: an empty set and a set with the element in it: `{{}, {a}}`
- If we're given a set with two elements in it (`{a, b}`), then the power is has four sets: `{{}, {a}, {b}, {a, b}}`

What's the pattern?

Notice that going from the empty set to `{a}`, that we still keep the empty set in our result and have another set with a in it. Similarly, when going from one element to two, we keep the same result set with one element (`{}, {a}`), but we also have a duplicate set with the b in it (`{b}, {a ,b}`).

So we can use the following recursive formula to generate the power set:

- If the input set is empty, return a set with an empty set in it
- Otherwise, take an element from our set. Let's call it `x`.
- Generate the power set of our input set without `x`. Let's call it `result`, for lack of a better name.
- Return the union of `name` with `name + x`

```
def power_set(s):
    if not s:
        return [[]]
    result = power_set(s[1:])
    return result + [subset + [s[0]] for subset in result]
```

This runs in O(2^N) time and space, since that's how many subsets there are.