Backtracking: The Power Set

William Fiset

What is the power set?

In mathematics, the power set is the set of all subsets. If s is a set we denote the power set as $\boldsymbol{P}(s)$.

Suppose that $s = \{a, b, c\}$

Then $\boldsymbol{P}(s) = \{\{\}, \{a\}, \{b\}, \{c\}, \{a,b\}, \{a,c\}, \{b,c\}, \{a,b,c\}\}$

Power set facts

P(s) always has 2^n elements where $n = |s|$

Power set facts

P(s) always has 2^n elements where $n = |s|$

The power set is the set of all subsets of different sizes. In other words, the power set is the set of all combinations of different sizes.

Subsets of size 0: {}

Subsets of size 1: {a}, {b}, {c}

Subsets of size 2: {a,b}, {a,c}, {b,c}

Subsets of size 3: {a,b,c}

Let's see how we can use backtracking to generate all subsets of a set. The key realization is to notice that **any subset can be represented as a bit string of length N**.

A B C

___ ___ ___

Let’s see how we can use backtracking to generate all subsets of a set. The key realization is to notice that **any subset can be represented as a bit string of length N**.

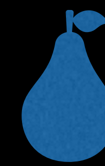
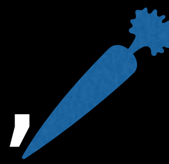
A	B	C
0	0	0

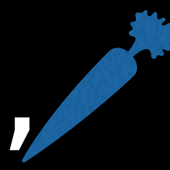
Let’s see how we can use backtracking to generate all subsets of a set. The key realization is to notice that **any subset can be represented as a bit string of length N**.

A	B	C
1	0	1

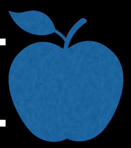
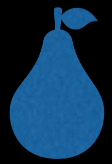
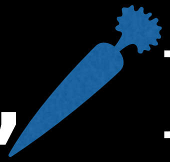
Let’s see how we can use backtracking to generate all subsets of a set. The key realization is to notice that **any subset can be represented as a bit string of length N**.

A	B	C
0	1	1

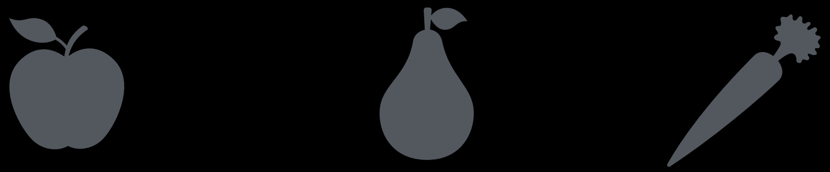
Suppose s = {, , }, what is **P**(s)?

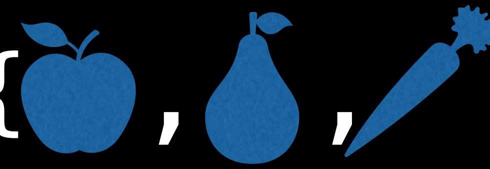
Suppose s = {, , }, what is **P**(s)?

0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

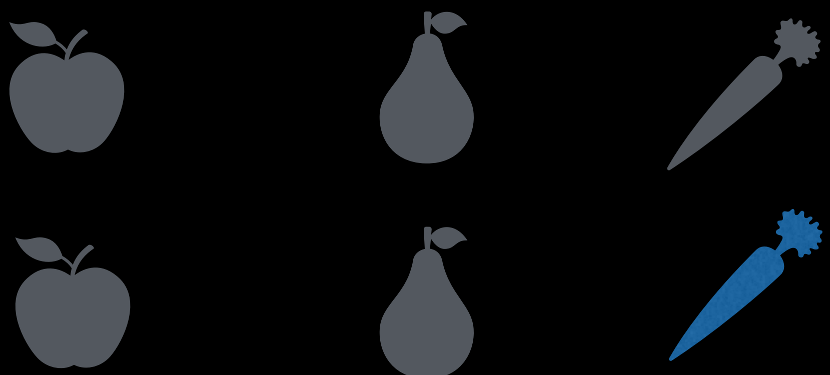
Suppose s = {,,}, what is **P**(s)?

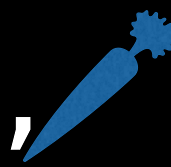
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1



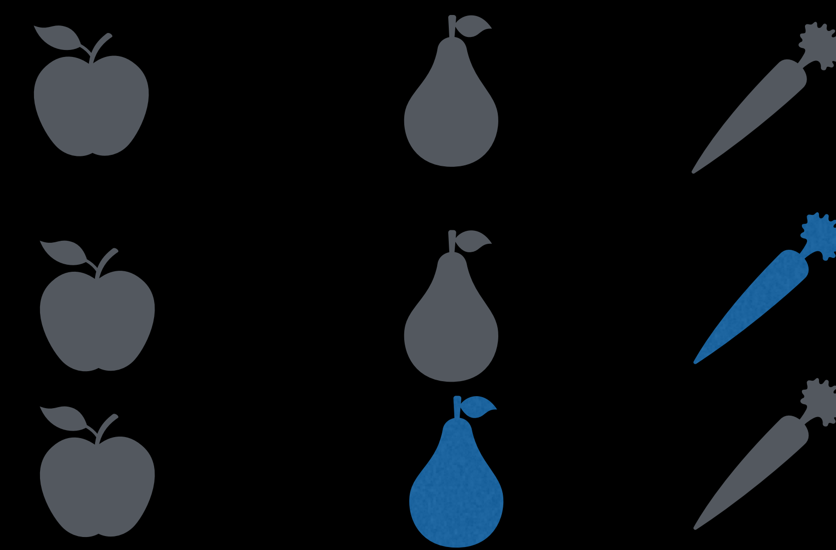
Suppose s = {}, what is **P**(s)?

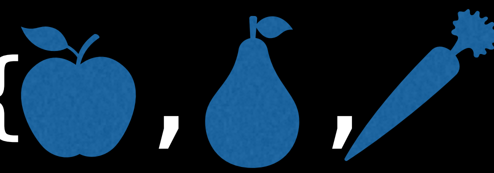
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1



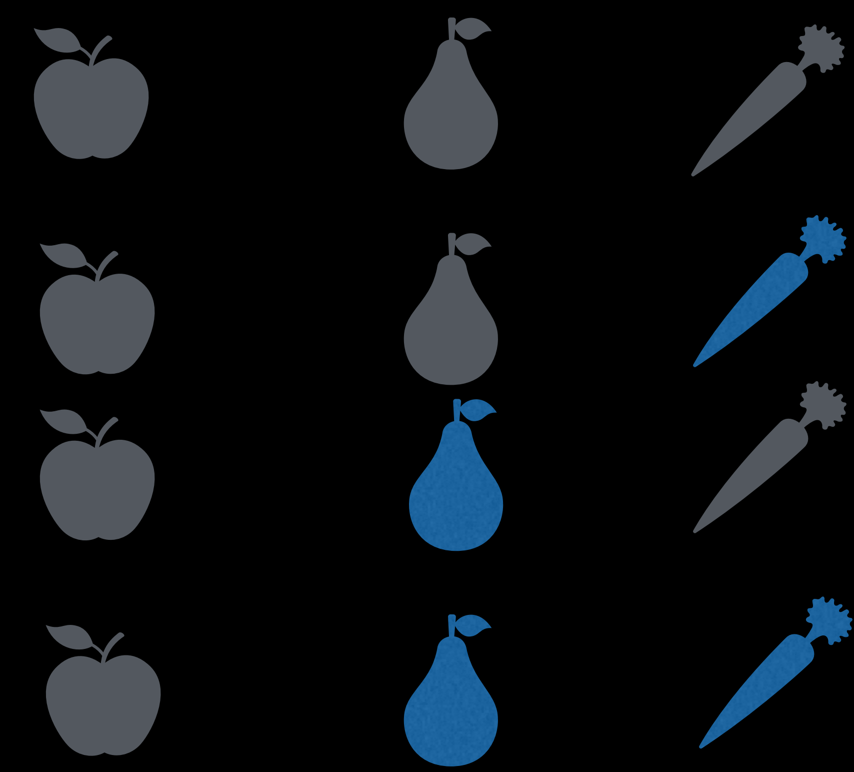
Suppose s = {,,}, what is **P**(s)?

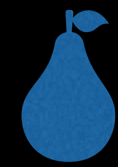
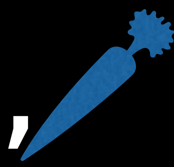
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1



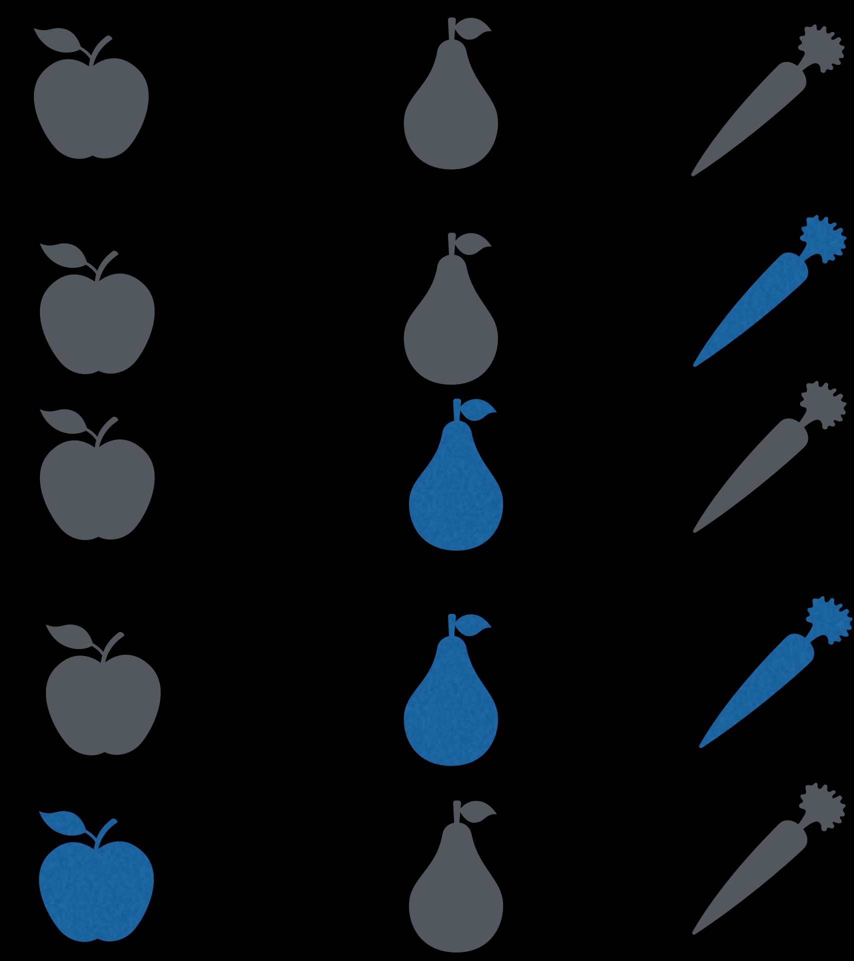
Suppose s = {}, what is **P**(s)?

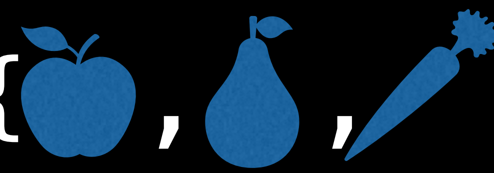
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1



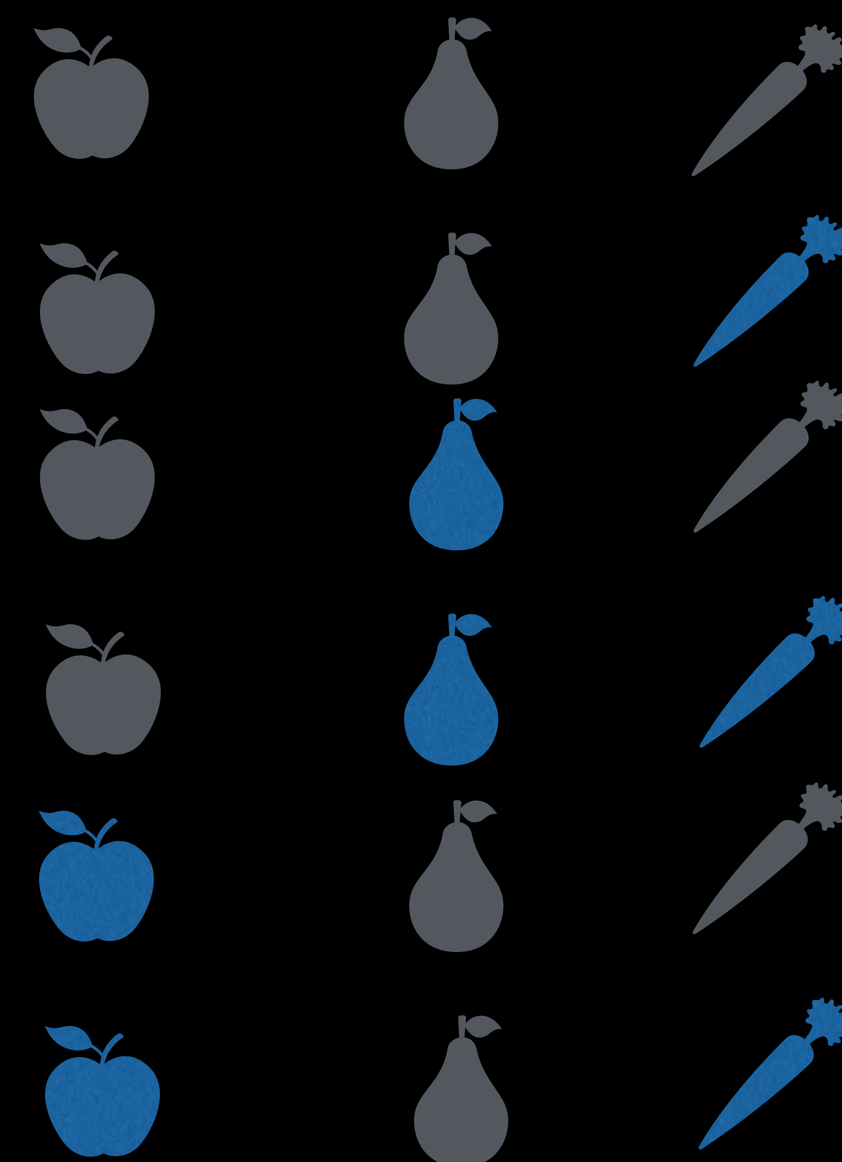
Suppose s = {,,}, what is $\mathbf{P}(s)$?

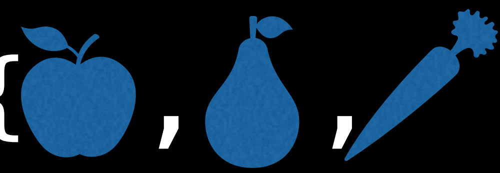
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1



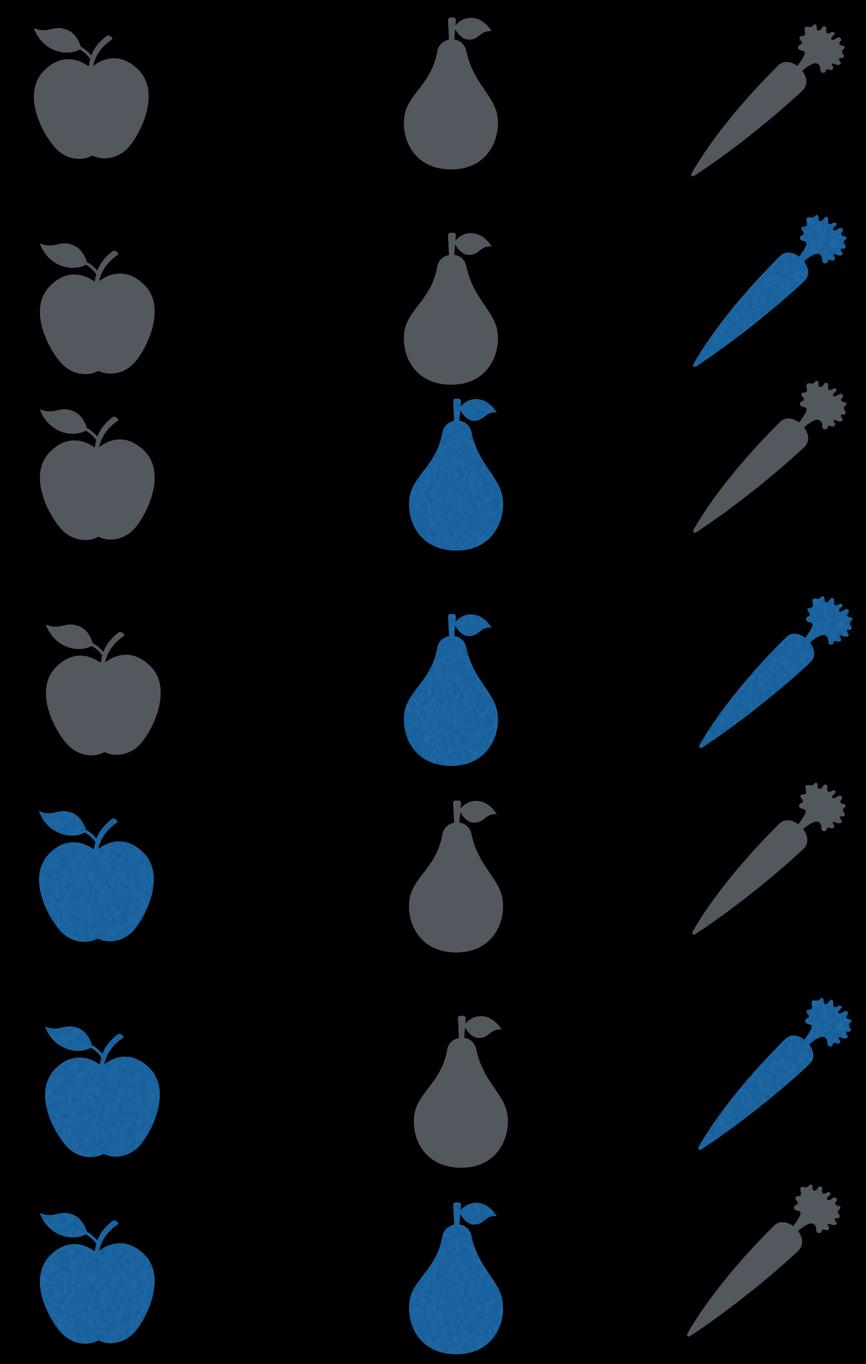
Suppose s = {}, what is **P**(s)?

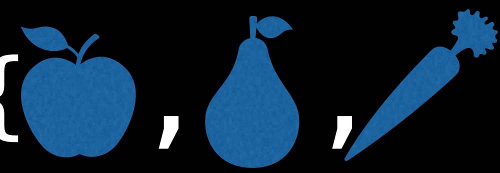
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1



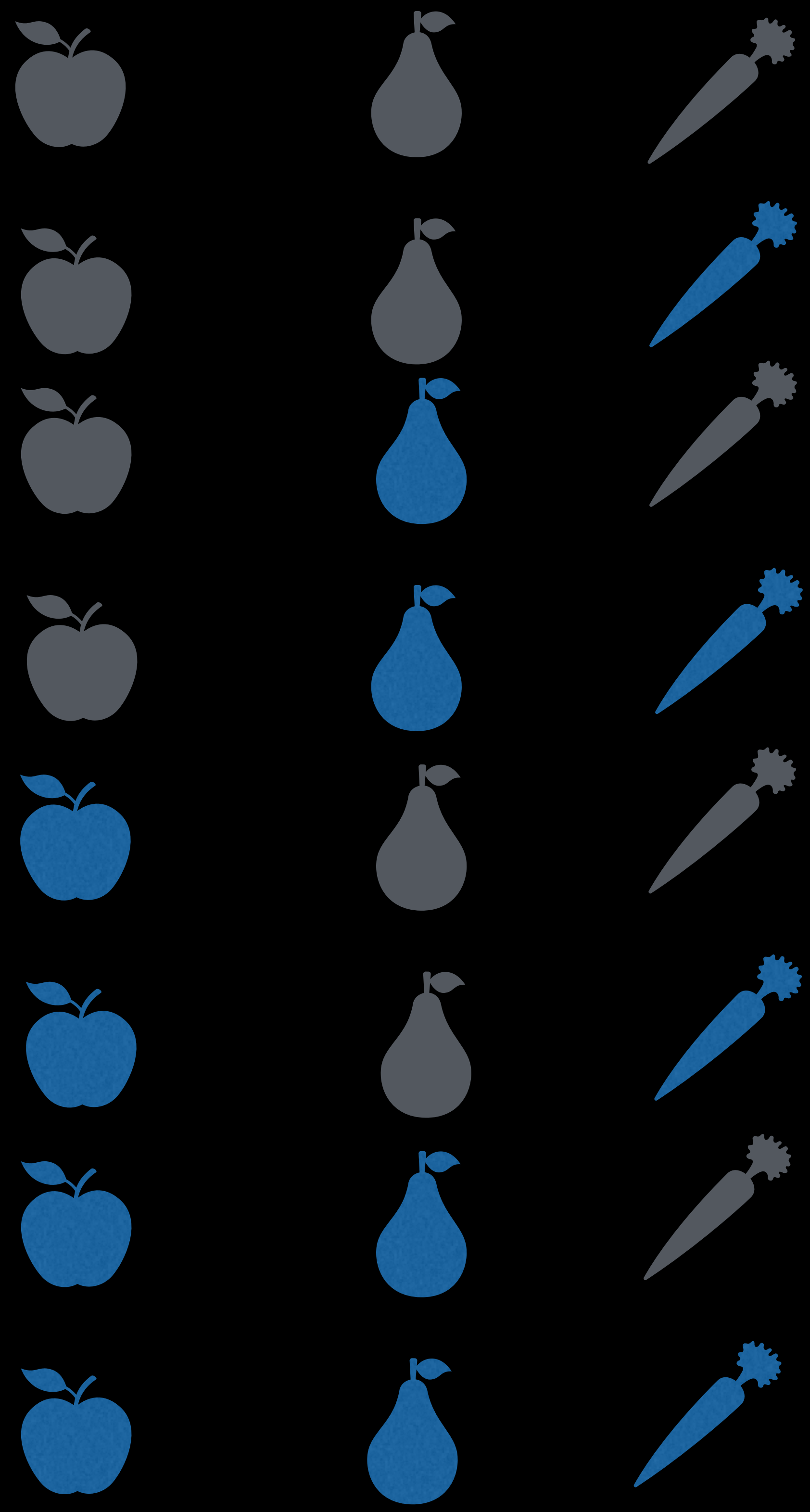
Suppose s = {}, what is **P**(s)?

0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1



Suppose s = {}, what is P(s)?

0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

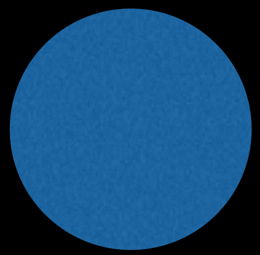


```
function powerSet(set):

    N = set.length
    B = [0,0,…,0] # B should be of length N
    bitstrings = []
    generateBitstrings(0, B, bitstrings)

    # Use found bit strings to select items
    subsets = []
    for bitstring in bitstrings:
        subset = []
        for (i = 0; i < N; i = i + 1)
            bit = bitstring[i]
            if bit == 1:
                subset.add(set[i])
        subsets.add(subset)
    return subsets
```

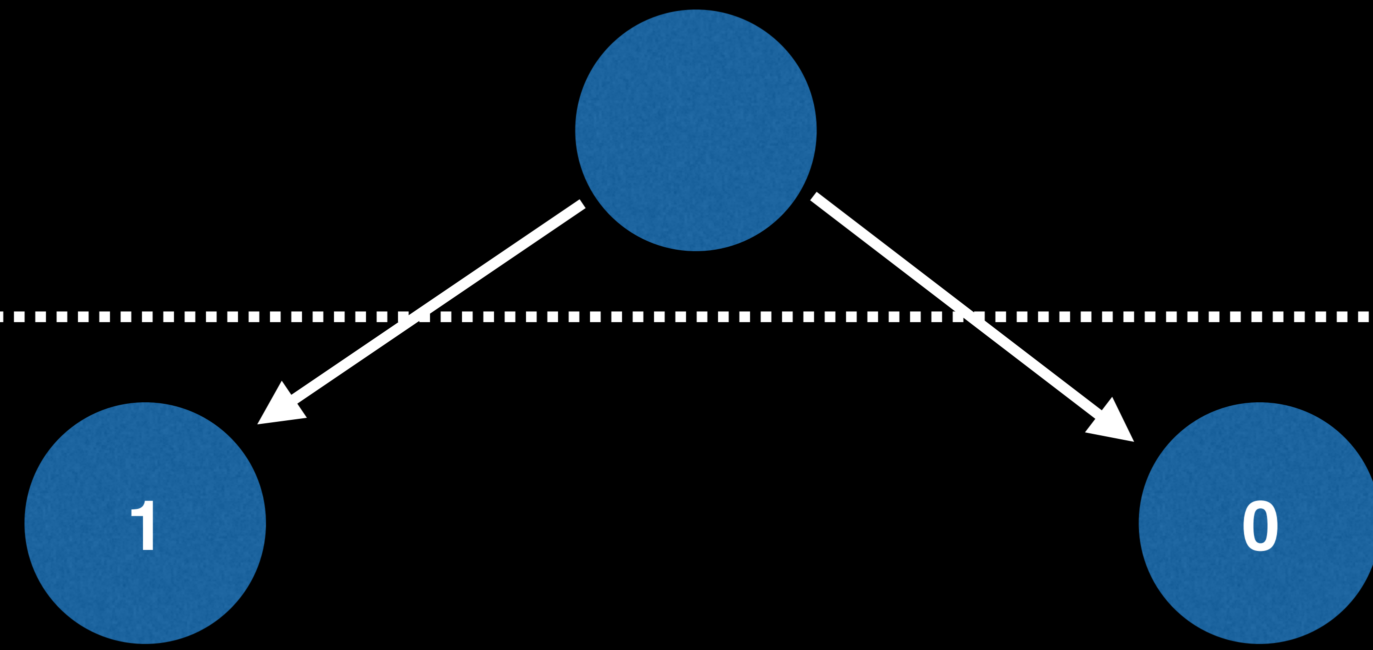
Let *i* be the index of the element we're considering, let *B* be an array of length *N* representing a bit string (initialized with all 0s), and finally, let *bitstrings* be an originally empty array tracking the found bit strings.

```
function generateBitstrings(i, B, bitstrings):

    # Found a valid subset
    if i == N
        bitstrings.add(B.deepcopy())
    else
        # Consider including element
        B[i] = 1
        generateBitstrings(i+1, B, bitstrings)

        # Consider not including element
        B[i] = 0
        generateBitstrings(i+1, B, bitstrings)
```

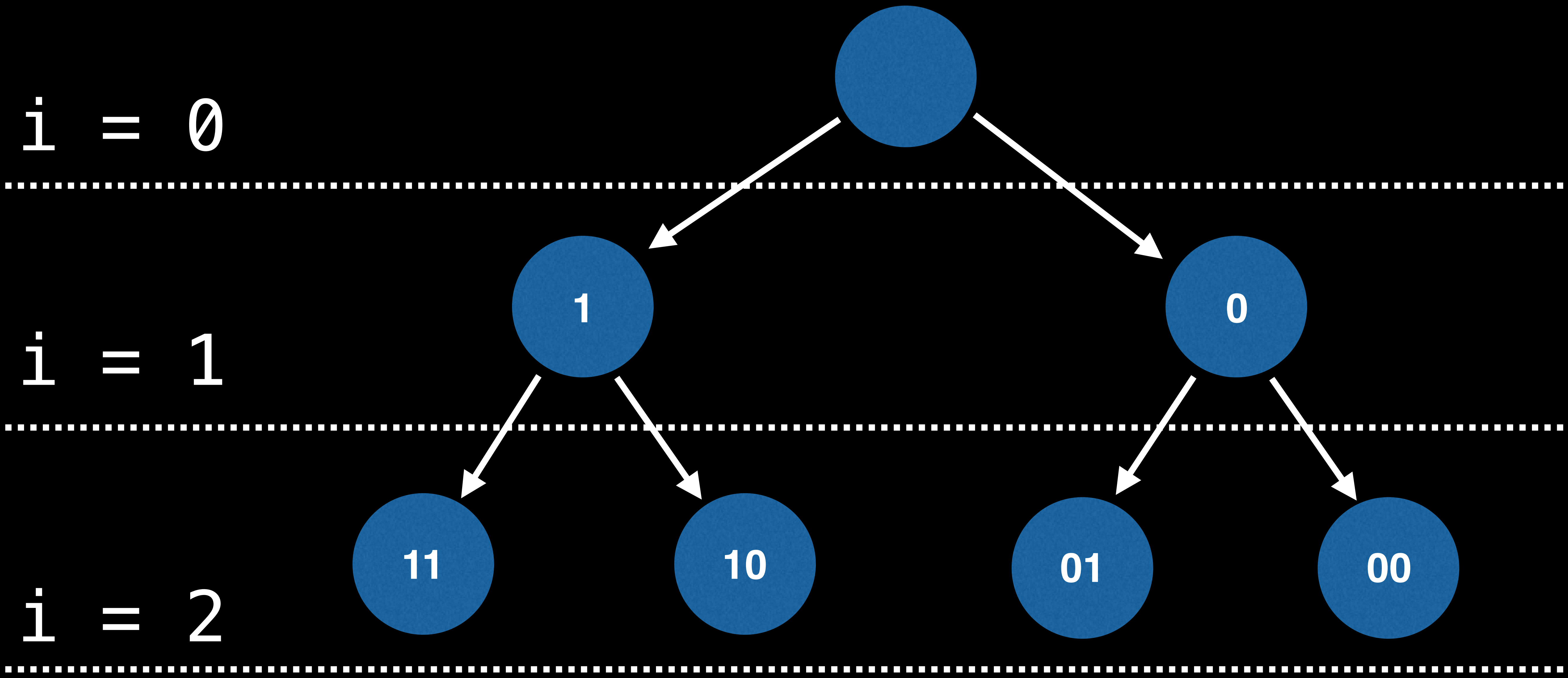
i = 0

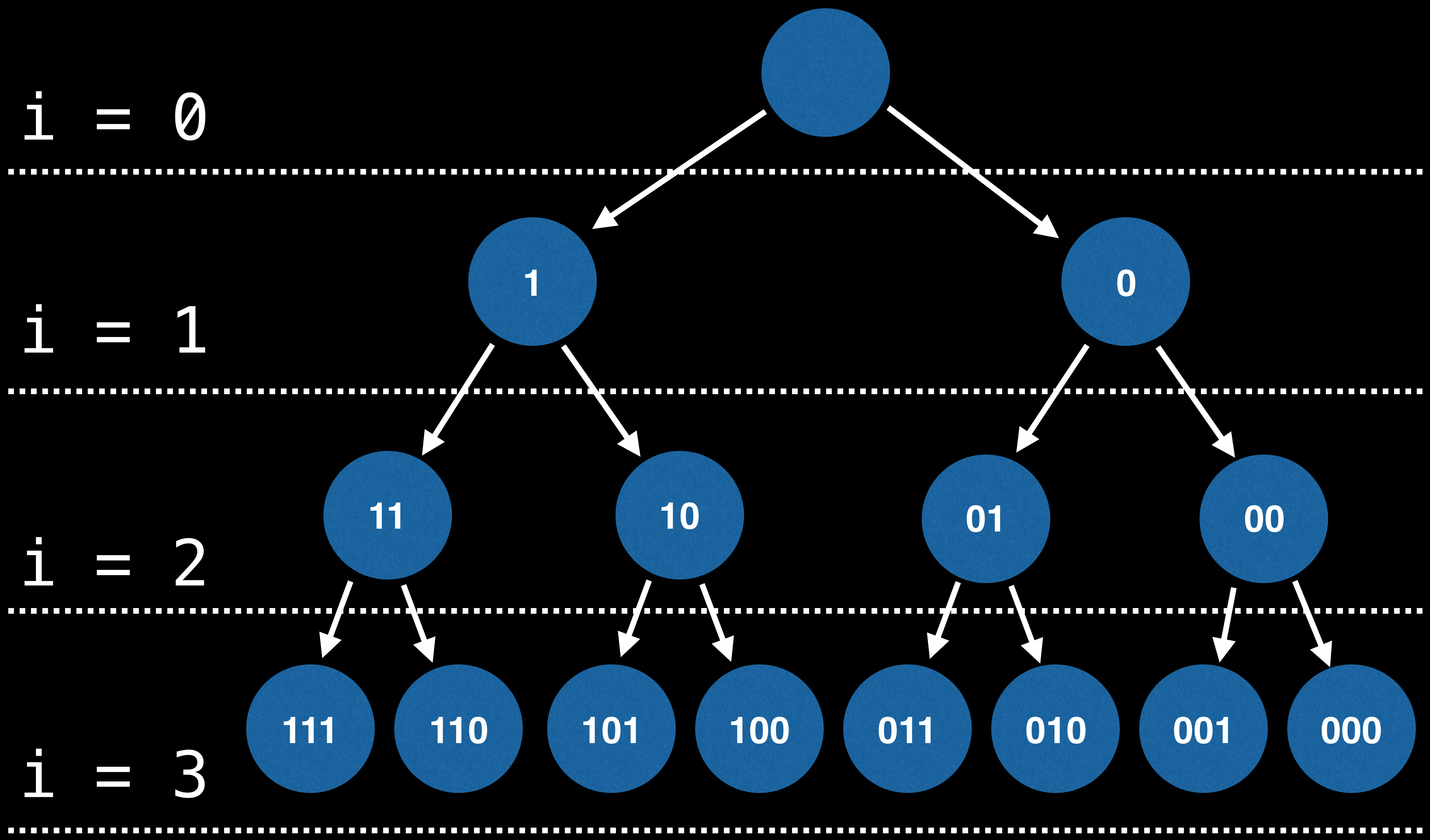


i = 0



i = 1


i = 0
i = 1
i = 2
1
0
11
10
01
00


i = 0
i = 1
i = 2
i = 3
1
0
11
10
01
00
111
110
101
100
011
010
001
000

Source Code Link

Implementation source code can be found at the following link:

github.com/williamfiset/algorithms

Link in the description: