SOS DP

SOS - Sum over Subset
What is Subset ?

I will be addressing the following problem: Given a fixed array **A** of $2^N$ integers, we need to calculate ∀ x function **F(x)** = Sum of all **A[i]** such that **x&i = i**, i.e., **i** is a subset of **x**.

$$F[mask] = \sum_{i \subseteq mask} A[i]$$

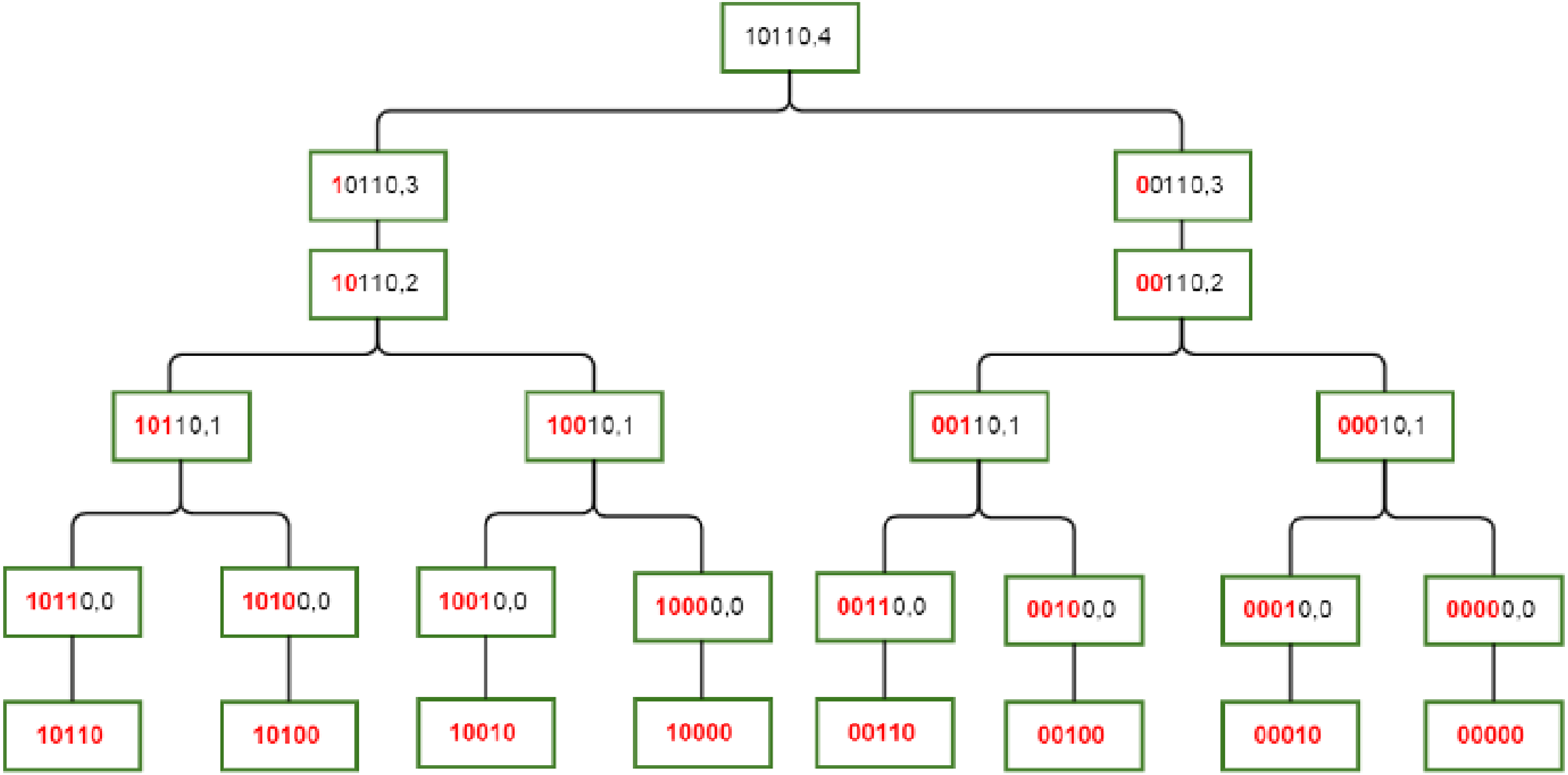
```
Input: A[] = {7, 12, 14, 16}  ,  n = 2
Output: 7, 19, 21, 49
Explanation: There will be 4 values of x: 0,1,2,3
So, we need to calculate F(0),F(1),F(2),F(3).
Now, F(0) = A0 = 7
F(1) =  A0 + A1 = 19
F(2) = A0 + A2 = 21
F(3) = A0 + A1 + A2 + A3 = 49

Input: A[] = {7, 11, 13, 16}  ,  n = 2
Output: 7, 18, 20, 47
Explanation: There will be 4 values of x: 0,1,2,3
So, we need to calculate F(0),F(1),F(2),F(3).
Now, F(0) = A0 = 7
F(1) =  A0 + A1 = 18
F(2) = A0 + A2 = 20
F(3) = A0 + A1 + A2 + A3 = 47
```

# Bruteforce

```
for(int mask = 0;mask < (1<<N); ++mask){
        for(int i = 0;i < (1<<N); ++i){
                if((mask&i) == i){
                        F[mask] += A[i];
                }
        }
}
```

This solution is quite straightforward and inefficient with time complexity of $O(4^N)$

$$S(mask,\, i) = \begin{cases} S(mask,\, i-1) & i^{th}\ bit\ OFF \\ S(mask,\, i-1) \bigcup S(mask \oplus 2^i,\, i-1) & i^{th}\ bit\ ON \end{cases}$$


10110,4
10110,3
00110,3
10110,2
00110,2
10110,1
10010,1
00110,1
00010,1
10110,0
10100,0
10010,0
10000,0
00110,0
00100,0
00010,0
00000,0
10110
10100
10010
10000
00110
00100
00010
00000

```
//iterative version
for(int mask = 0; mask < (1<<N); ++mask){
	dp[mask][-1] = A[mask]; //handle base case separately (leaf states)
	for(int i = 0;i < N; ++i){
		if(mask & (1<<i))
			dp[mask][i] = dp[mask][i-1] + dp[mask^(1<<i)][i-1];
		else
			dp[mask][i] = dp[mask][i-1];
	}
	F[mask] = dp[mask][N-1];
}
```

```
//memory optimized, super easy to code.
for(int i = 0; i<(1<<N); ++i)
	F[i] = A[i];
for(int i = 0;i < N; ++i) for(int mask = 0; mask < (1<<N); ++mask){
	if(mask & (1<<i))
		F[mask] += F[mask^(1<<i)];
}
```

The above algorithm runs in $O(N\,2^N)$ time.