Algorithm 1 sumSubSet

Input : array[], n, K
Output : memoi[n][K]

```
let memoi[n + 1][K + 1] be new table
for i = 0 to n do
    memoi[i][0] = TRUE
end for
for i = 1 to K do
    memoi[0][i] = FALSE
end for
for i = 1 to K do
    for j = 1 to n do
        if arrray[j − 1] > i then
            memoi[j][i] = memoi[j − 1][i]
        else if arrray[j − 1] ≤ i then
            memoi[j][i] = memoi[j − 1][i−array[j − 1]|| memoi[j − 1][i]
        end if
return memoi[n][K]
```

새로운 memoi[n+1][K+1] 배열을 생성하여 계산 결과를 저장한다. 먼저 memoi의 첫 번째 row와 col을 초기화한다. 이때 col은 True, row는 False로 초기화한다.

memoi[j][i]의 값은 array[j-1]의 값과 i를 비교하여 판단한다. 첫 번째로 array[j-1]의 값이 i보다 큰 경우에는 array[0]부터 array[j-1]까지의 값을 더하면 i의 값보다 큰 경우가 되므로 array[j-1]이 제외되고 array[0]부터 array[j-2]까지의 값의 합의 결과가 i인 memoi[j-1][i]의 값과 같게 된다. 두 번째로 array[j-1]의 값이 i보다 작거나 같은 경우 array[0]부터 array[j-1]의 합이 i와 같을 수 있는 경우가 되므로 이 경우에는 memoi[j-1][i - array[j-1]]과 memoi[j-1][i]의 값 중 True인 경우 memoi[j][i]가 True가 된다. 아래 그림은 예시를 나타내었다.

	0	1	2	3	4	5	6	7	8
0	True	False	False	False	False	False	False	False	False
1	True	True	False	False	False	False	False		
3	True	True	False	True	True	False	False		
6	True	True	False	True	True	False			
7	True	True	False	True	True	False			
11	True	True	False	True	True	False			
16	True	True	False	True	True	False			

그림 2

[그림 2] 처럼 Bottom-Up DP방식으로 채워나간다.

시간복잡도 $T(n)$을 생각해보자. 위 pseudo code에서 보면 for loop에 의해서
$T(n) = O(n) + O(n) + O(nK) \leq O(nK)$이므로 $T(n) = O(nK)$이다.