부분집합 생성하기

- 각 원소가 부분집합에 포함되었는지를 loop 이용하여 확인하고 부분집합을 생성하는 방법

```
bit = [0, 0, 0, 0]
for i in range(2) :
    bit[0] = i                  # 0번째 원소
    for j in range(2) :
        bit[1] = j              # 1번째 원소
        for k in range(2) :
            bit[2] = k          # 2번째 원소
            for l in range(2) :
                bit[3] = l      # 3번째 원소
                print(bit)      # 생성된 부분집합 출력
```

비트 연산자

비트 연산자

&	비트 단위로 AND 연산을 한다.
|	비트 단위로 OR 연산을 한다.
<<	피연산자의 비트 열을 왼쪽으로 이동시킨다.
>>	피연산자의 비트 열을 오른쪽으로 이동시킨다.

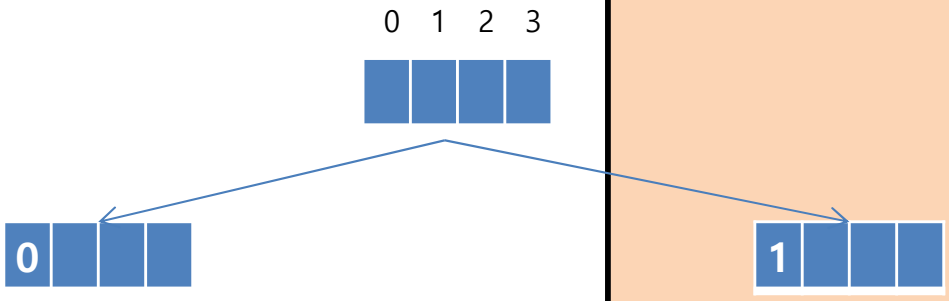
<< 연산자

- 1 << n: 2^n 즉, 원소가 n개일 경우의 모든 부분집합의 수를 의미한다.

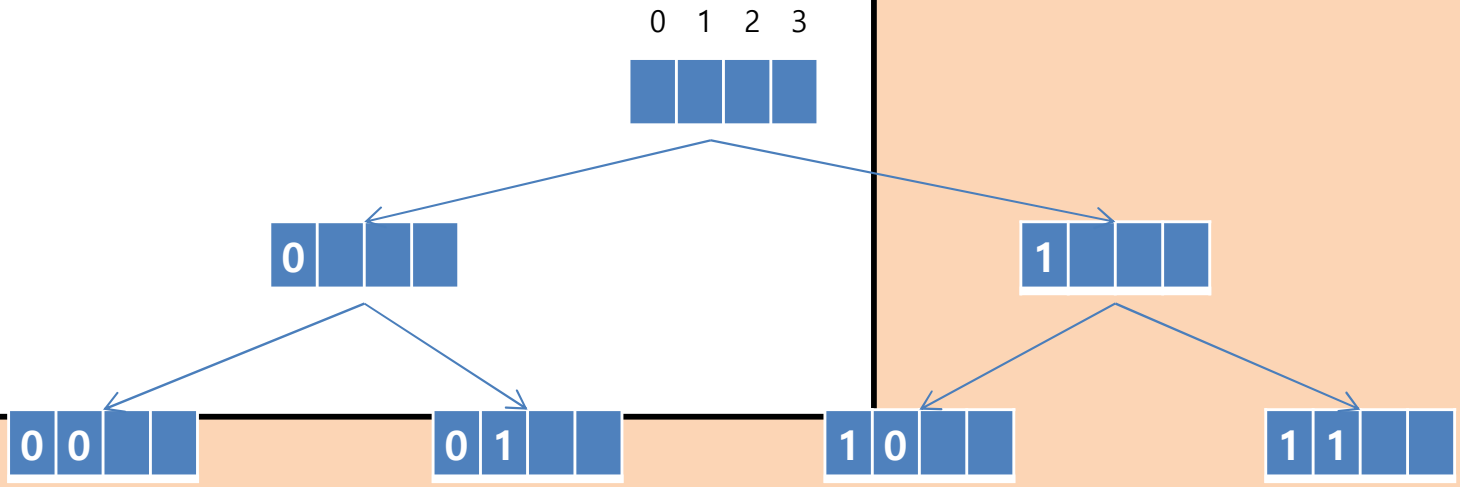
& 연산자

- i & (1<<j) : i의 j번째 비트가 1인지 아닌지를 리턴한다.

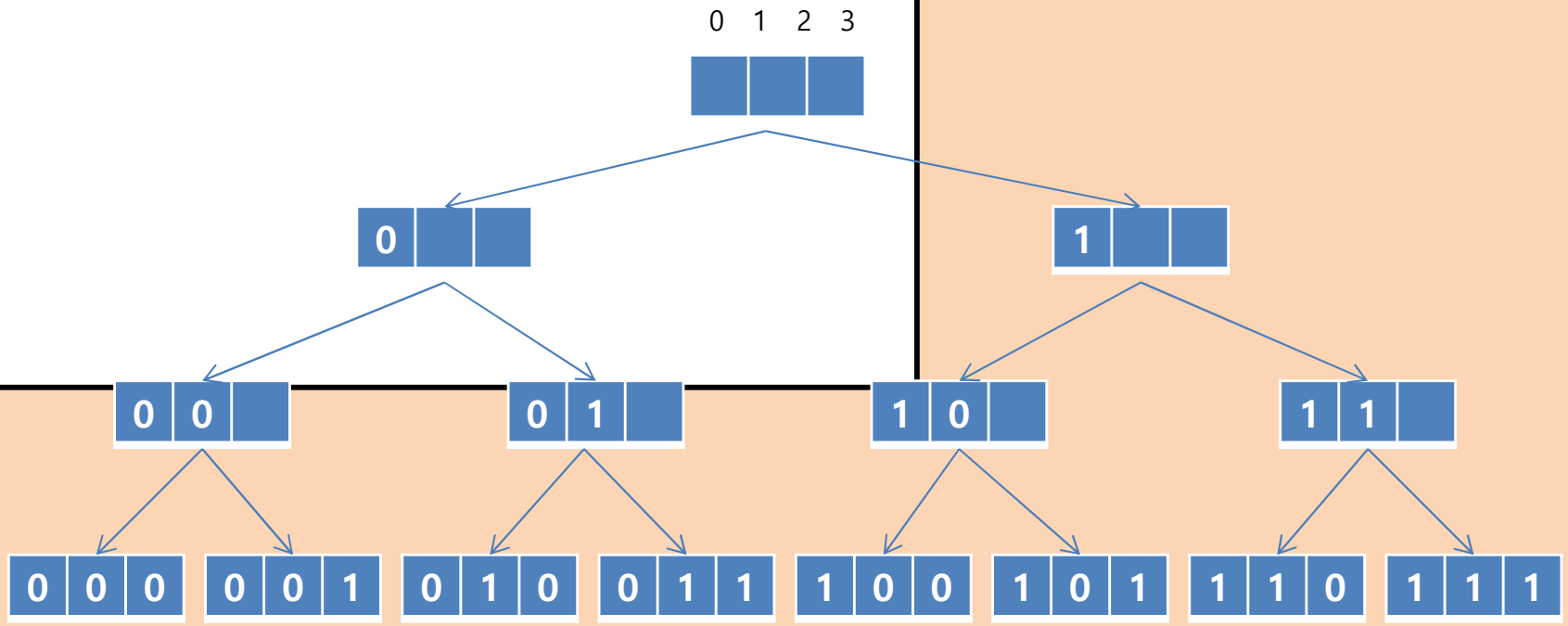
```
for i from 0 to 1
    bit[0] ← (i%2==0)?0:1;
```



```
for i from 0 to 1 {
    bit[0] ← (i%2==0)?0:1;
    for j from 0 to 1 {
        bit[1] ← (j%2==0)?0:1;
```



```
for i from 0 to 1
    bit[0] ← (i%2==0)?0:1;
    for j from 0 to 1
        bit[1] ← (j%2==0)?0:1;
        for k from 0 to 1
            bit[2] ← (k%2==0)?0:1;
```

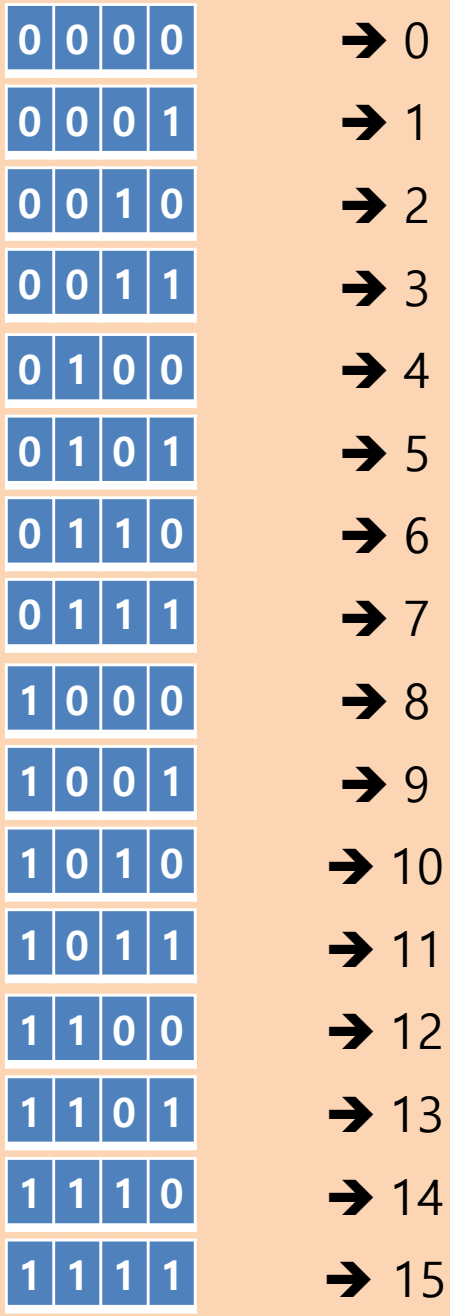


```
for i from 0 to 1
    bit[0] ← (i%2==0)?0:1;
    for j from 0 to 1
        bit[1] ← (j%2==0)?0:1;
        for k from 0 to 1
            bit[2] ← (k%2==0)?0:1;
                print_array();
```

0	1	2	
0	0	0	{ }
0	0	1	{ 3 }
0	1	0	{ 2 }
0	1	1	{ 2, 3 }
1	0	0	{ 1 }
1	0	1	{ 1, 3 }
1	1	0	{ 1, 2 }
1	1	1	{ 1, 2, 3 }

0	1	2
1	2	3

bit로 생각
해 보면...


0 0 0 0 → 0
0 0 0 1 → 1
0 0 1 0 → 2
0 0 1 1 → 3
0 1 0 0 → 4
0 1 0 1 → 5
0 1 1 0 → 6
0 1 1 1 → 7
1 0 0 0 → 8
1 0 0 1 → 9
1 0 1 0 → 10
1 0 1 1 → 11
1 1 0 0 → 12
1 1 0 1 → 13
1 1 1 0 → 14
1 1 1 1 → 15

```
for i in range(1<<n) :
    for j in range(n):
        if i & (1<<j):
            print(arr[j], end=", ")
```

n = 3

0	1	2

0	1	2
1	2	3

0	0	0	{ }
0	0	1	{ 1 }
0	1	0	{ 2 }
0	1	1	{ 1, 2 }
1	0	0	{ 3 }
1	0	1	{ 1, 3 }
1	1	0	{ 2, 3 }
1	1	1	{ 1, 2, 3 }