백준 n queens 문제 (비트마스킹 안쓰면 시간 초과 남)

```python
import sys; reader = sys.stdin.readline
def search(col, ld, rd, n):
    size = ((1 << n) - 1)
    count = 0

    if col == size:
        return 1

    slots = ~(col | ld | rd) & size
    while slots:
        bit = slots & -slots
        count += search(col | bit, (ld | bit) >> 1, (rd | bit) << 1, n)
        slots -= bit
    return count


def solution(n):
    return search(0, 0, 0, n)
n = int( reader().rstrip() )
print(solution(n))
```

경우의 수 30-31 가지 이상의 상황에서는 비트마스킹을 활용 불가 **(2^30 = 1,073,741,824)** 펜윅트리에서 비트마스킹이 유용

AND	**&** (1 & 1 == 1)
OR	| (1 | 0 == 1) (1 | 1 == 1)
XOR	^ (1 | 0 == 1) (1 | 1 == 0)
NOT	~ (~1010 == 0101)
[SHIFT] 1 << num	1 << 3 == 1000
[SHIFT] num >> 1	2 >> 10111 == 101

활용-예시) 하나의 피자에 선택/취소 할 수 있는 토핑 20 가지가 존재

공집합	bit = 0	피자 토핑 20 개 모두 선택 안함
꽉찬 비트 집합	bit = (1 << 20) - 1	피자 토핑 20 개 모두 선택
비트 추가	bit |= (1 << number)	특정 토핑 추가
비트 삭제	bit &= (1 << number)	특정 토핑 취소하기
비트 토글	bit ^= (1 << number)	특정 토핑을 이미 선택했으면 취소하고 선택하지 않았으면 추가하기
켜져있는 최하위 비트 구하기	smallestbit = bit & -bit	보수 개념 활용
최하위 비트 삭제	bit &= (bit - 1)	
모든 비트 순회	for i in range(1<<비트자리수) : for j in range(1<<비트자리수): if i & (1 << j) : print("exist")	

예시) 두 집합 A 와 B 의 연산

합집합	result = Abit | Bbit
교집합	result = Abit & Bbit
차집합 (A-B)	result = Abit | ~Bbit
(A - B) **U** (B - A)	result = Abit ^ Bbit
집합 크기 구하기	def bitSize (bit) : if bit == 0 : return 0 ret = bit % 2 + bitSize(bit // 2) return ret

모든 부분 집합 순회	i = bit
	while i :
	  print(i)
	  i = (i - 1) & bit

사용 빈도 높은 연산들

idx 번째 비트 끄기	bit &= ~ (1<< idx)
idx 번째 비트 켜기	bit |= (1 << idx)
idx 번째 비트 토글	bit ^= (1 << idx)
idx 번째 비트 유무 확인	if bit & (1 << idx) : print("EXIST")
최하위 켜져있는 idx 찾기	idx = (bit & -bit)
크기가 n 인 집합의 모든 비트 켜기	bit = (1 << n) - 1