비트마스크(Bit Mask)

비트마스크(Bitmask)

비트(bit)연산을 사용해서 부분 집합을 표현할 수 있다.
&(and), | (or), ~(not), ^(xor)

A	B	~A	A & B	A | B	A ^ B
0	0	1	0	0	0
0	1	1	0	1	1
1	0	0	0	1	1
1	1	0	1	1	0

- ~ : 반전
- & : 두 비트가 모두 1일 때 1
- | : 두 비트 중 하나가 1일 때 1
- ^ : 두 비트가 다를 때 1

```
  0 0 1 1 0 1 1      0 0 1 1 0 1 1      0 0 1 1 0 1 1
& 1 0 1 0 0 1 1    | 1 0 1 0 0 1 1    ^ 1 0 1 0 0 1 1
---------------    ---------------    ---------------
  0 0 1 0 0 1 1      1 0 1 1 0 1 1      1 0 0 1 0 0 0
```

Gaussian

비트마스크(Bitmask)

- shift left(<<) 와 shift right(>>) 연산

A << B (A를 왼쪽으로 B비트만큼 이동)
☞ $A * 2^B$

1 << 0 = 1
1 << 1 = 2 (10_2)
1 << 2 = 4 (100_2)
1 << 3 = 8 (1000_2)
3 << 3 = 24 (1100_2)

A >> B (A를 오른쪽으로 B비트만큼 이동)
☞ $A / 2^B$

1 >> 0 = 1
1 >> 1 = 0
10 >> 1 = 5 (101_2)
10 >> 3 = 1 (1_2)

- (A + B) / 2 는 (A + B) >> 1로 쓸 수 있다. (이분탐색 때 사용 가능)
- 어떤 수가 홀 수 인지 판별하는 if (N % 2 == 1)은 if (N & 1)로 줄여 쓸 수 있다.

- 정수로 집합을 나타낼 수 있다.
- $\{1, 3, 4, 5, 9\} = 2^1 + 2^3 + 2^4 + 2^5 + 2^9 = 1000111010_2$
  ☞ 길이가 N인 2진수로 표현할 수 있다.

비트마스크(Bitmask)

▣ 포함 검사

{1, 3, 4, 5, 9} = $2^1 + 2^3 + 2^4 + 2^5 + 2^9 = 1000111010_2$ = 570

- 0이 포함되어 있는지 검사
  : 570 & 2^0 = 570 & (1<<0) = 0
- 1이 포함되어 있는지 검사
  : 570 & 2^1 = 570 & (1<<1) = 2
- 3이 포함되어 있는지 검사
  : 570 & 2^3 = 570 & (1<<3) = 8

▣ 비트 추가 연산

{1, 3, 4, 5, 9} = 570

- 1 추가하기
  : 570 | 2^1 = 570 | (1<<1) = 570 (1000111010_2)
- 2 추가하기
  : 570 | 2^1 = 570 | (1<<2) = 574 (1000111110_2)

비트마스크(Bitmask)

▣ 비트 제거 연산

{1, 3, 4, 5, 9} = 570

- 1 제거하기

  : 570 & ~2^1 = 570 & ~(1<<1) = 568 (1000111000_2)

	1	0	0	0	1	1	1	0	1	0
&	1	1	1	1	1	1	1	1	0	1
	1	0	0	0	1	1	1	0	0	0

2^1 삭제

- 2 제거하기

  : 570 | 2^1 = 570 ~ (1<<2) = 570 (1000111010_2)

	1	0	0	0	1	1	1	0	1	0
&	1	1	1	1	1	1	1	0	1	1
	1	0	0	0	1	1	1	0	1	0

삭제 없음

- 전체 집합 : (1 << N) -1
- 공집합 : 0

비트마스크(Bitmask)

▣ 프로그래밍 시 사용 할 것

현재 집합이 S일 때

① i를 추가 : S | (1<<i)

② i를 제거 : S & ~(1 << i)

③ i를 검사 : S & (1 << i)

④ i를 토글(0 → 1, 1 → 0) : S ^ (1<<i)

집합

https://www.acmicpc.net/problem/11723

☞ **집합에 원소를 추가/제거/검사/토글**

- 비트마스크를 사용하는 이유는 집합을 배열의 인덱스로 표현할 수 있기 때문
- 상태 다이나믹을 할 때 자주 사용하게 된다.