비트마스크

위승빈 (Winning-Bean)

비트마스크란?

정수의 이진수로 표현하고, 비트 연산을 통해 문제를 해결하는 기법

컴퓨터는 내부적으로 모든 자료를 이진수로 표현한다. 이런 특성을 이용해 정수의 이진수 표현을 자료구조로 쓰는 기법을 말한다.

예를 들어 {false, true, true, false, false, true, true, true}; 는 01100111로 나타낼 수 있다. 이렇게 비트마스크로 나타낸다면 아래와 같은 장점이 있다.

- 비트 연산을 통해 삽입, 삭제, 조회가 간단
- 메모리 사용량을 줄임
- 간결한 코드 작성
- 더욱 빠른 연산
- 다이나믹 프로그래밍 가능

또한, 비트마스킹을 이용하면 집합을 쉽게 표현 할 수 있다.

비트 연산

AND 연산 (&)

a & b : 대응 수 다 1 이라면 1, 아니면 0

01100111 & 10110110 = 00100110

OR 연산 (|)

a | b : 대응 수 다 0 이라면 0, 아니면 1

01100111 | 10110110 = 11110111

XOR 연산 (^)

a ^ b : 대응 수 서로 다르면 1, 아니면 0

01100111 ^ 10110110 = 11010001

LEFT SHIFT 연산 (<<)

a << b : a를 b비트만큼 왼쪽으로 밀어낸다.

01100111 << 2 = 10011100

RIGHT SHIFT 연산 (>>)

a >> b : a를 b비트만큼 오른쪽으로 밀어낸다.

01100111 >> 2 = 00011001

NOT 연산 (~)

~a : 비트 값 반전

~01100111 = 10011000

비트마스크로 집합 표현하기

비트가 1이면 원소가 포함, 비트가 0이면 원소가 불포함으로, 집합을 비트마스크로 표현

예를 들어 {1, 2, 3, 4, 5} 집합에서 {1} 부분집합은 10000, {3, 4} 부분집합은 00110으로 표현할 수 있다.

원소 추가

a | (1 << n) : n번째 수를 추가 할 때, 현재 상태(a)에서 n번 비트를 1로 바꿔준다.

{3, 4} 부분집합에 1 추가 (즉, 오른쪽에서 0부터 시작하여 4번째) : 00110 | (1 << 4) = 10110

원소 삭제

a & (1 << n) : n번째 수를 제거 할 때, 현재 상태(a)에서 n번 비트를 0으로 바꿔준다.

{1, 3, 4} 부분집합에 1 제거 (즉, 오른쪽에서 0부터 시작하여 4 index) : 10110 & ~(1 << 4) = 00110

원소 조회

a & ~(1 << n) : n번째 수의 존재 유무를 확인 할 때, 0이면 없고 1이상이면 있는 것이다.

{1, 2, 3, 4, 5} 집합에서 오른쪽에서 0 index 확인 (현재 {3, 4}) : 00110 & (1 << 0) = 00000 (없음)

{1, 2, 3, 4, 5} 집합에서 오른쪽에서 2 index 확인 (현재 {3, 4}) : 00110 & (1 << 2) = 00100 (있음)

원소 토글

a ^ (1 << n) : n번째 수가 있으면 없애고 없으면 있도록 만든다.

{3, 4} 부분집합에 오른쪽에서 0 index 토글 : 00110 ^ (1 << 0) = 00111

비트마스크로 집합끼리 연산하기

a | b : a와 b의 합집합

a & b : a와 b의 교집합

a & ~b : a에서 b를 뺀 차집합

a ^ b : a와 b 중 하나에만 포함된 원소들의 집합