## 데이터 업데이트가 가능한 상황에서의 구간 합 (Interval Sum) 문제

- **BOJ '구간 합 구하기' 문제**: https://www.acmicpc.net/problem/2042
- 어떤 N개의 수가 주어져 있다. 그런데 중간에 수의 변경이 빈번히 일어나고 그 중간에 어떤 부분의 합을 구하려 한다. 만약에 1, 2, 3, 4, 5라는 수가 있고, 3번째 수를 6으로 바꾸고 2번째부터 5번째까지 합을 구하라고 한다면 17을 출력하면 되는 것이다. 그리고 그 상태에서 다섯 번째 수를 2로 바꾸고 3번째부터 5번째까지 합을 구하라고 한다면 12가 될 것이다.
- **데이터 개수**: N (1 ≤ N ≤ 1,000,000)
- **데이터 변경 횟수**: M (1 ≤ M ≤ 10,000)
- **구간 합 계산 횟수**: K (1 ≤ K ≤ 10,000)
- 이 문제를 어떻게 해결할 수 있을까요?



## 바이너리 인덱스 트리 (Binary Indexed Tree)

- **바이너리 인덱스 트리(binary indexed tree)**는 2진법 인덱스 구조를 활용해 구간 합 문제를 효과적으로 해결해 줄 수 있는 자료구조를 의미합니다.
  - **펜윅 트리(fenwick tree)**라고도 합니다.
- 정수에 따른 2진수 표기

| 정수 | 2진수 표기 |
|---|---|
| 7 | 00000000 00000000 00000000 0000111 |
| -7 | 11111111 11111111 11111111 1111001 |

- **0이 아닌 마지막 비트를 찾는 방법**
  - 특정한 숫자 K의 0이 아닌 마지막 비트를 찾기 위해서 **K & -K**를 계산하면 됩니다.

## 바이너리 인덱스 트리 (Binary Indexed Tree)

- K & -K 계산 결과 예시

| 정수 K | 2진수 표기 | K & -K |
|---|---|---|
| 0 | 00000000 00000000 00000000 00000000 | 0 |
| 1 | 00000000 00000000 00000000 00000001 | 1 |
| 2 | 00000000 00000000 00000000 00000010 | 2 |
| 3 | 00000000 00000000 00000000 00000011 | 1 |
| 4 | 00000000 00000000 00000000 00000100 | 4 |
| 5 | 00000000 00000000 00000000 00000101 | 1 |
| 6 | 00000000 00000000 00000000 00000110 | 2 |
| 7 | 00000000 00000000 00000000 00000111 | 1 |
| 8 | 00000000 00000000 00000000 00001000 | 8 |



## 바이너리 인덱스 트리 (Binary Indexed Tree)

- K & -K 계산 결과 예시

```python
n = 8

for i in range(n + 1):
    print(i, "의 마지막 비트:", (i & -i))
```

실행 결과

```
0 의 마지막 비트: 0
1 의 마지막 비트: 1
2 의 마지막 비트: 2
3 의 마지막 비트: 1
4 의 마지막 비트: 4
5 의 마지막 비트: 1
6 의 마지막 비트: 2
7 의 마지막 비트: 1
8 의 마지막 비트: 8
```

## 바이너리 인덱스 트리: 트리 구조 만들기

- **트리 구조 만들기**: 0이 아닌 마지막 비트 = 내가 저장하고 있는 값들의 개수

| 인덱스 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 ~ 16 | | | | | | | | | | | | | | | |
| | 1 ~ 8 | | | | | | | | 9 ~ 16 | | | | | | | |
| | 1 ~ 4 | | | | | | | | 9 ~ 12 | | | | | | | |
| | 1 ~ 2 | | | | 5 ~ 6 | | | | 9 ~ 10 | | | | 13 ~ 14 | | | |
| | 1 | | 3 | | 5 | | 7 | | 9 | | 11 | | 13 | | 15 | |

바이너리 인덱스 트리 (Binary Indexed Tree)

## 바이너리 인덱스 트리: 업데이트 (Update)

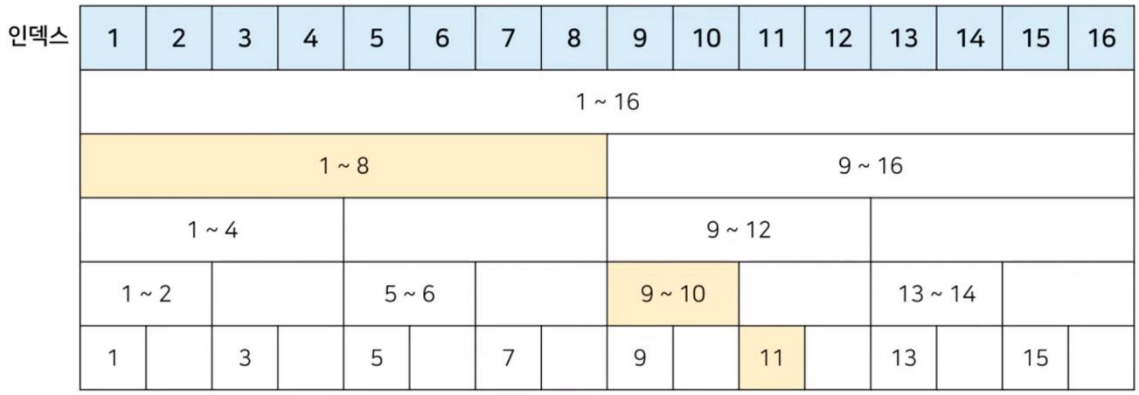
- **특정 값을 변경할 때**: 0이 아닌 마지막 비트만큼 더하면서 구간들의 값을 변경 (예시 = 3rd)

| 인덱스 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 ~ 16 | | | | | | | | | | | | | | | |
| | 1 ~ 8 | | | | | | | | 9 ~ 16 | | | | | | | |
| | 1 ~ 4 | | | | | | | | 9 ~ 12 | | | | | | | |
| | 1 ~ 2 | | | | 5 ~ 6 | | | | 9 ~ 10 | | | | 13 ~ 14 | | | |
| | 1 | | 3 | | 5 | | 7 | | 9 | | 11 | | 13 | | 15 | |

바이너리 인덱스 트리 (Binary Indexed Tree)

## 바이너리 인덱스 트리: 누적 합 (Prefix Sum)

- **1부터 N까지의 합(누적 합) 구하기**: 0이 아닌 마지막 비트만큼 빼면서 구간들의 값의 합 계산 (예시 = 11th)






## 바이너리 인덱스 트리 구현: 파이썬(Python)

```python
import sys
input = sys.stdin.readline

# 데이터의 개수(n), 변경 횟수(m), 구간 합 계산 횟수(k)
n, m, k = map(int, input().split())

# 전체 데이터의 개수는 최대 1,000,000개
arr = [0] * (n + 1)
tree = [0] * (n + 1)

# i번째 수까지의 누적 합을 계산하는 함수
def prefix_sum(i):
    result = 0
    while i > 0:
        result += tree[i]
        # 0이 아닌 마지막 비트만큼 빼가면서 이동
        i -= (i & -i)
    return result

# i번째 수를 dif만큼 더하는 함수
def update(i, dif):
    while i <= n:
        tree[i] += dif
        i += (i & -i)
```

```python
# start부터 end까지의 구간 합을 계산하는 함수
def interval_sum(start, end):
    return prefix_sum(end) - prefix_sum(start - 1)

for i in range(1, n + 1):
    x = int(input())
    arr[i] = x
    update(i, x)

for i in range(m + k):
    a, b, c = map(int, input().split())
    # 업데이트(update) 연산인 경우
    if a == 1:
        update(b, c - arr[b]) # 바뀐 크기(dif)만큼 적용
        arr[b] = c
    # 구간 합(interval sum) 연산인 경우
    else:
        print(interval_sum(b, c))
```

바이너리 인덱스 트리 (Binary Indexed Tree)