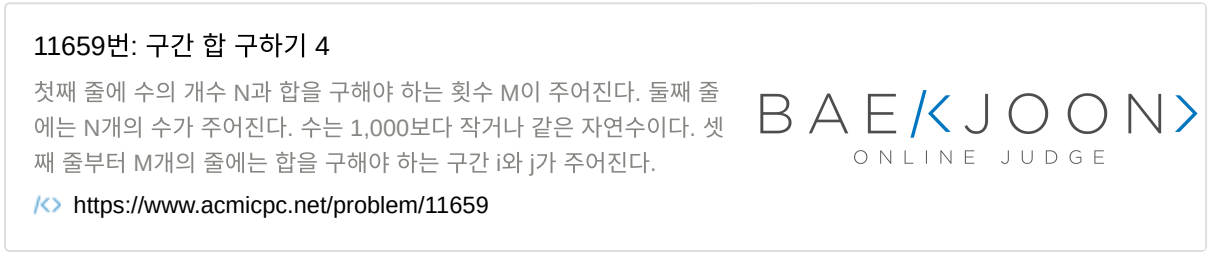
13주차 : Segment Tree

> Segment tree는 구간의 정보를 기록하는 tree이다.

필요성 

기본 문제

- 구간의 합 반복해서 구하기

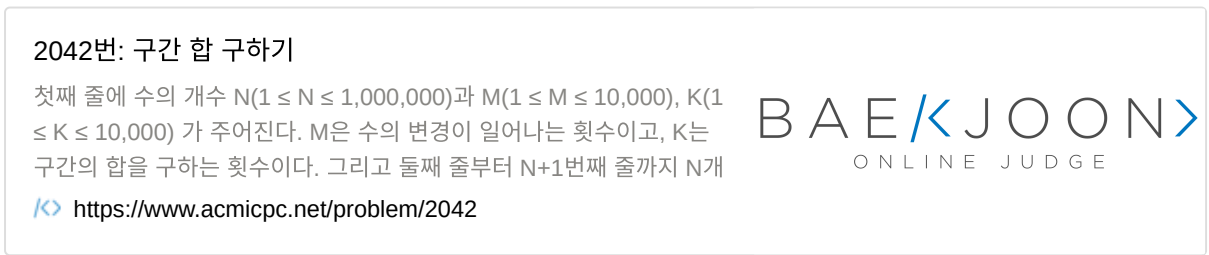


- 시간복잡도(길이: n, 구간 합 연산 횟수: m)
  - 부분 합 배열 생성: O(n)
  - 구간 합 연산: O(1) * m
  - 총: O(n+m)

ref. https://hroad.tistory.com/52

문제 상황

- 구간의 일부가 반복해서 변하는 경우



- 시간복잡도(길이: n, 구간 합 연산 횟수: m, 값 변경 횟수: k)
  - 기본 문제의 구간 합 방식
    - 부분 합 배열 생성: O(n)
    - 구간 합 연산: O(1) * m
    - 변경: O(n) * k
    - 총: O(n + m + **nk**) → **TLE**

Segment Tree 🏝️

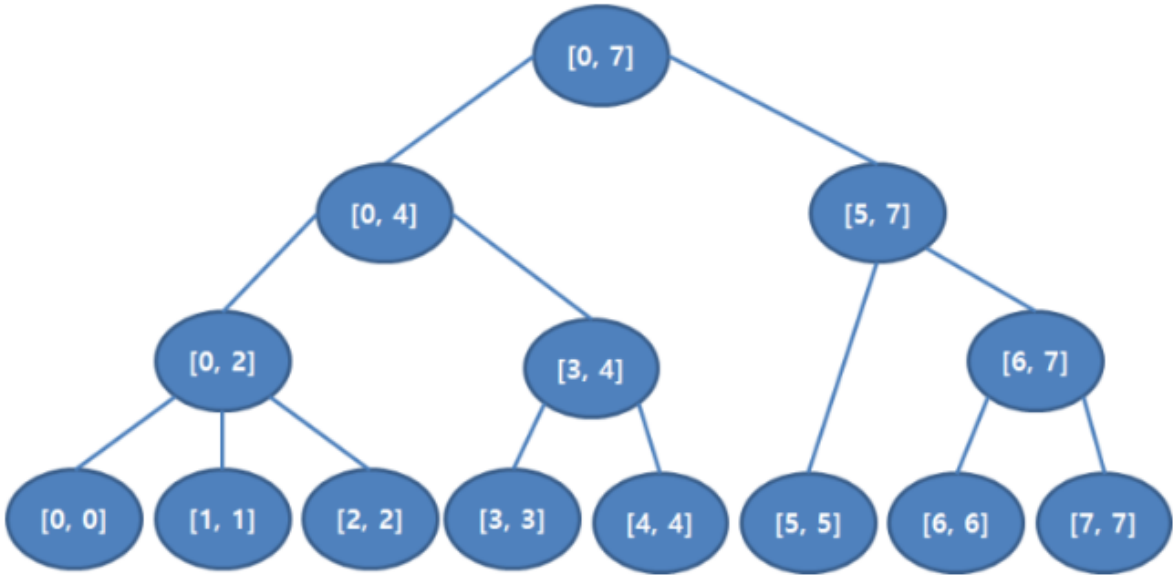
> Segment tree는 구간의 정보를 기록하는 tree이다.

구간의 정보?

- **구간의 합**
- 구간의 곱
- 구간의 최댓값
- 구간의 최솟값
- …

트리 구조

- 각 노드는 특정 범위를 의미
- 부모 노드는 자식 노드의 범위를 완벽하게 포함

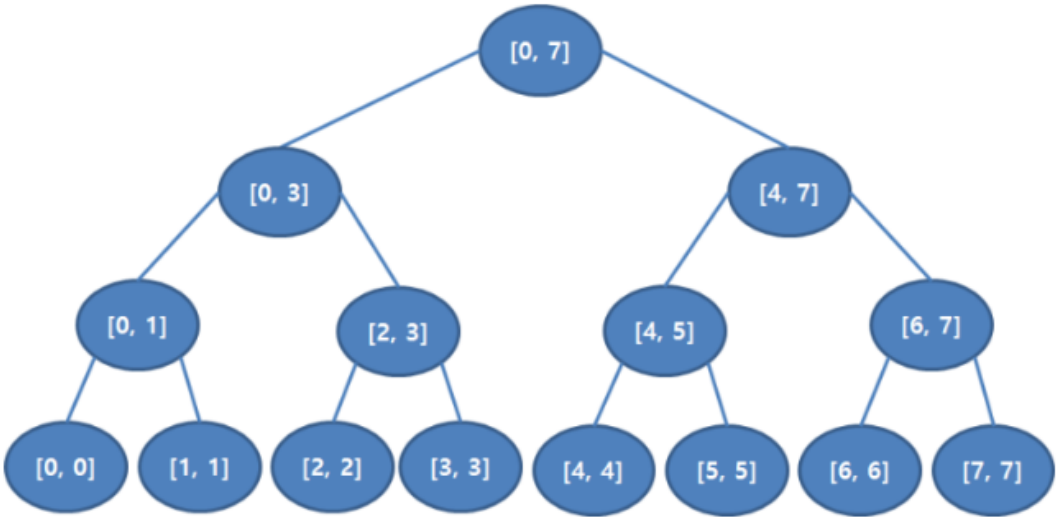


ref. https://m.blog.naver.com/PostView.naver?blogId=kks227&logNo=220791986409&navType=by

- a 위치의 단일 값을 [a, a] 범위로 생각하여 리프 노드에 배치 가능
- 루트 노드는 전체 구간을 의미
- 연속적인 구간을 가짐

일반적 사용

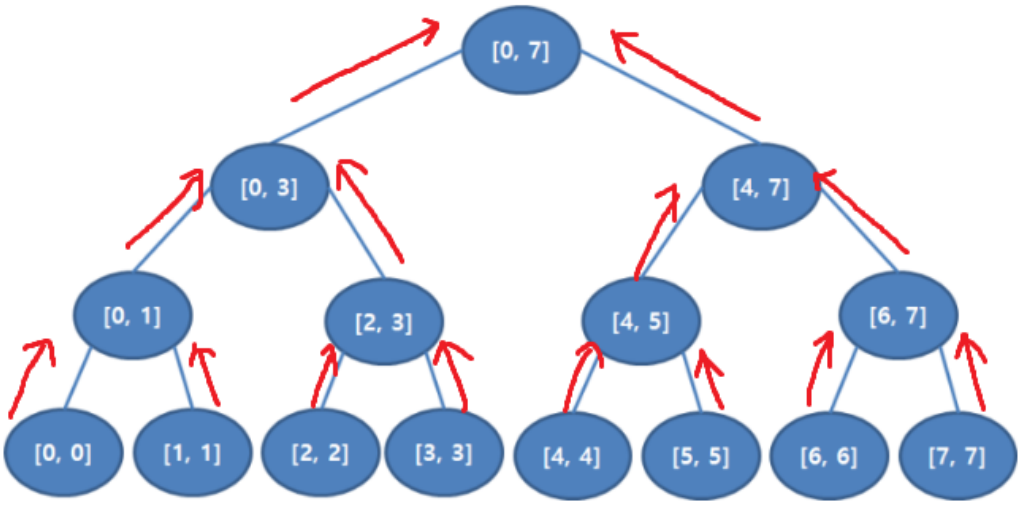
- 포화 이진 트리의 형태를 사용



- 갯수 n이 2 ** k가 아닌 경우에도 2 ** (k-1) < n < 2 ** k 를 만족하는 깊이가 k인 포화 이진 트리를 사용, 남는 리프 노드는 빈 값으로 채워서 사용

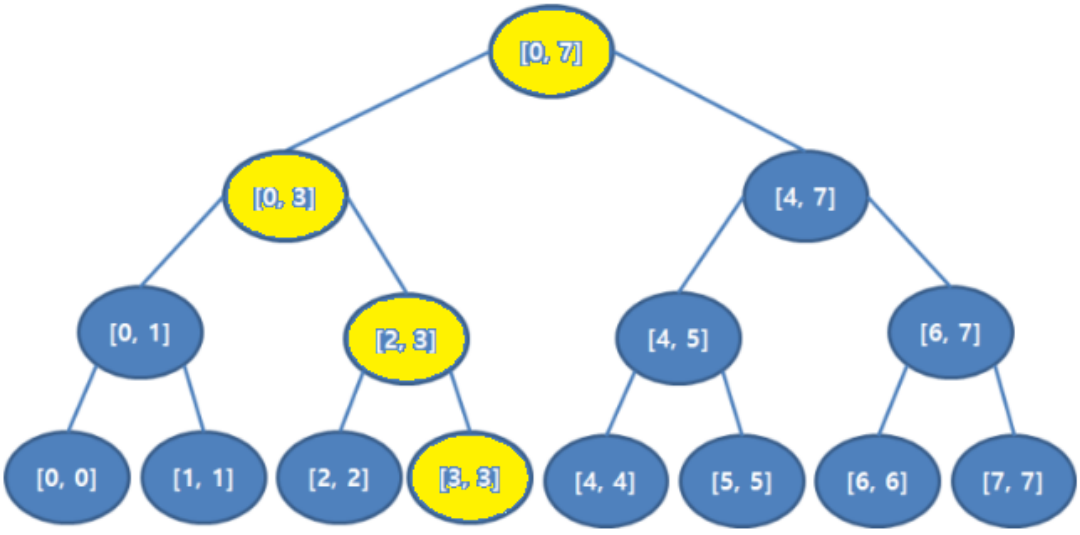
트리 생성

- 포화 이진 트리 생성(그냥 리스트 쓰면 됨)
- 리프 노드에 값 넣기
- 리프 노드에서부터 루트 노드까지
    - 자식 노드를 더해 부모 노드에 넣기

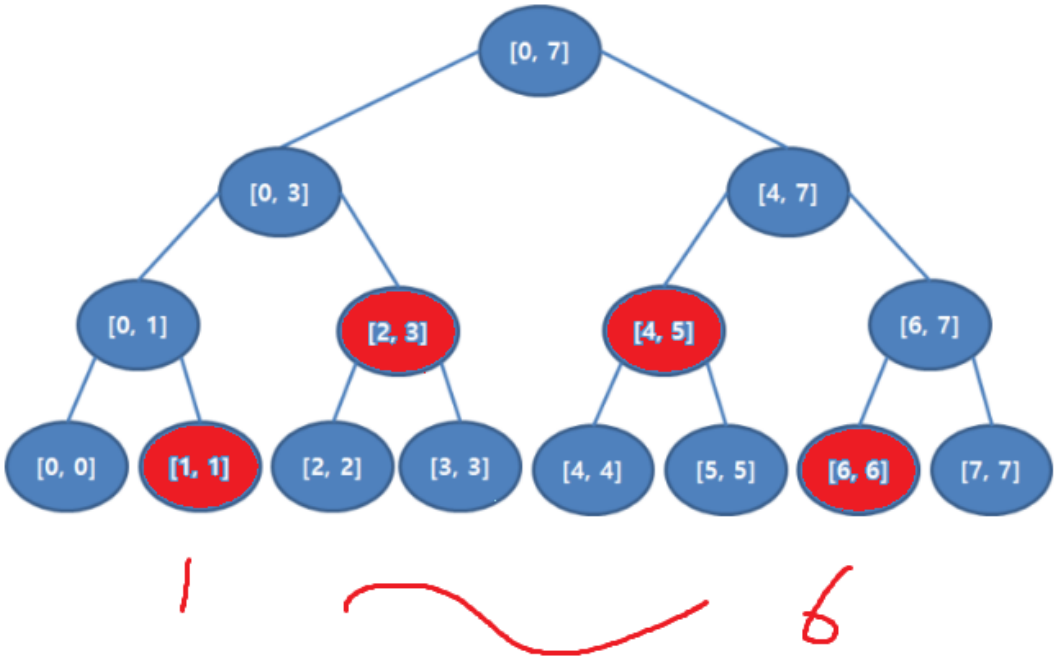


변경

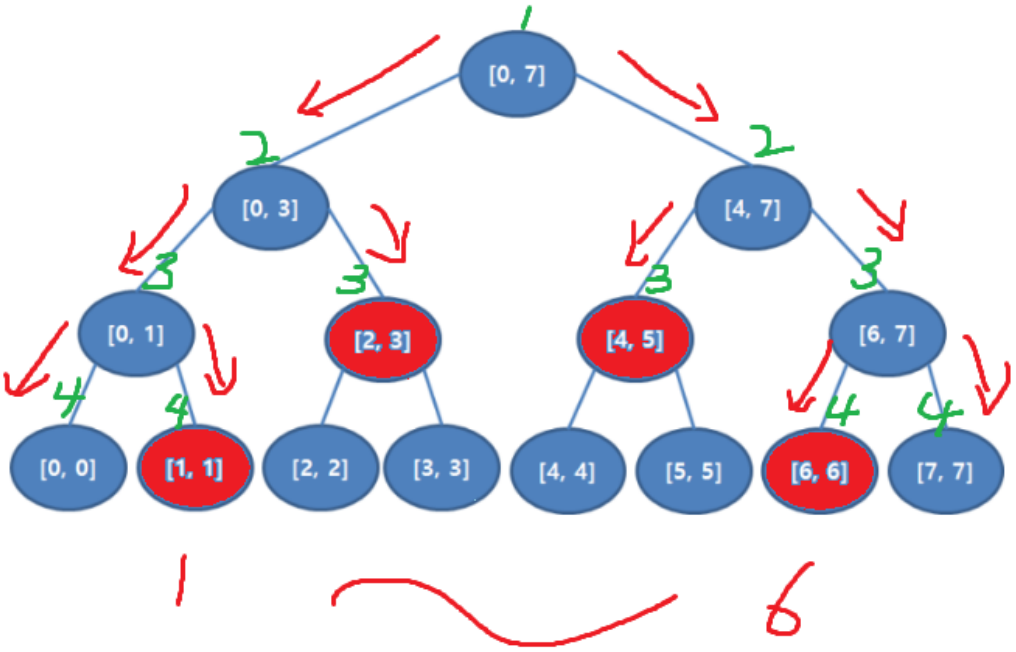
- 리프 노드를 변경
- 루트 노드에 도달할 때까지 부모 노드를 변경



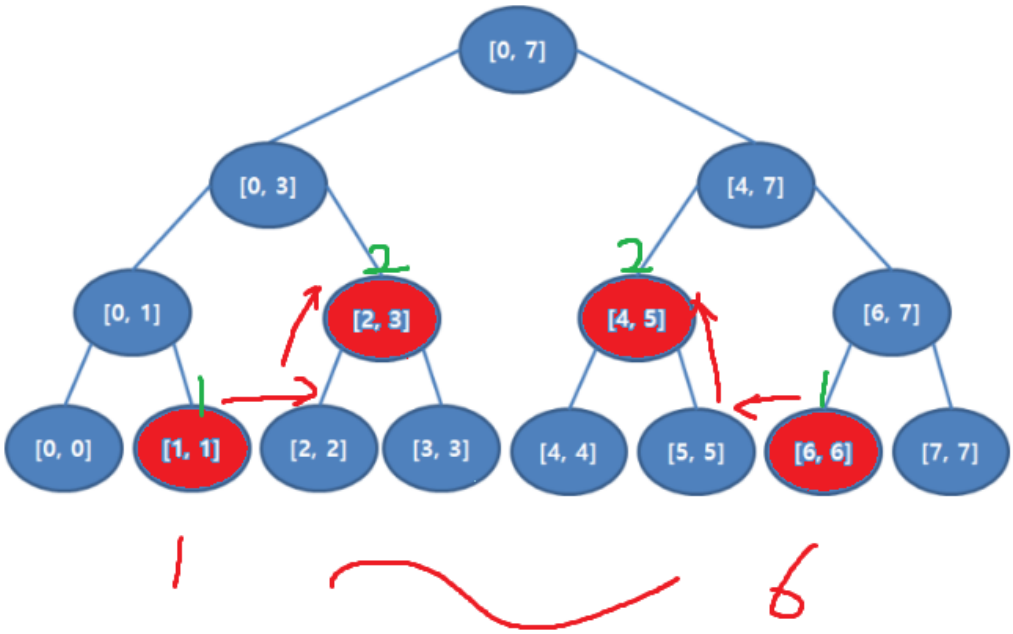
구간 합 계산



1. 루트 노드 → 리프 노드
    - 루트 노드에서 시작
    - 재귀
        - 노드의 범위가 구간 내에 포함되면
            - 구간 합에 더함
        - 노드의 범위가 구간 내에 포함되지 않으면
            - 두 개의 자식 노드 확인(재귀)



2. 리프 노드 → 루트 노드
    - 비트마스킹
    - 시작 노드 ≤ 종료 노드인 경우 반복
        - 시작 노드가 부모 노드의 오른쪽 노드라면
            - 구간 합에 더함
            - 시작 노드의 인덱스 + 1
        - 종료 노드가 부모 노드의 왼쪽 노드라면
            - 구간 합에 더함
            - 종료 노드의 인덱스 - 1
        - 시작 노드와 종료 노드를 각각의 부모 노드로 변경



시간 복잡도

길이: n, 구간 합 연산 횟수: m, 값 변경 횟수: k

- 트리 생성: O(n)
- 구간 합 연산: O(log(n)) * m
- 변경: O(log(n)) * k
- 총: O(n + (m+k)*log(n))

오늘의 문제

- 구간 합 구하기(Gold 1): https://www.acmicpc.net/problem/2042
- 세그먼트 트리의 생성, 변경, 구간 합 계산 알고리즘을 이용해 문제를 풀어보자!
- 루트 → 리프 방식이 구현이 쉽다.
- 해설이나 참고 자료를 참고하자!

▼ 해설(이호형, 루트 → 리프 방식)

```
import sys


def make_tree(tree):
    # 그냥 인덱스 역순으로 돌면서 자식 노드 두 개 더해서 부모 노드에 넣기
    for idx in range(len(tree)-2, 0, -2):
        tree[idx//2] = tree[idx] + tree[idx+1]
```

```
def change(tree, idx, num):
    diff = num - tree[idx]
    # 리프 노드에서 루트 노드로 이동하며 값 변경
    while idx >= 0:
        tree[idx] += diff
        idx = (idx-1) // 2  # 부모 노드로 변경


def summation(tree, start, end, now, left, right):
    # 노드의 구간이 계산 구간에 포함된 경우 더하기
    if start <= left and right <= end:
        return tree[now]
    # 노드의 구간이 계산 구간에서 완전히 벗어난 경우 prunning
    if right < start or left > end:
        return 0
    # 노드의 구간이 계산 구간에 걸쳐 있는 경우 자식 노드 두 개 재귀 호출
    return summation(tree, start, end, (now << 1) + 1, left, (left+right)//2) + \
           summation(tree, start, end, (now << 1) + 2, (left+right)//2 + 1, right)


CHANGE = 1
SUMMATION = 2
input = sys.stdin.readline
N, M, K = map(int, input().split())

# 트리 깊이 구하기
n = N
for cnt in range(N):
    n = n >> 1
    if n == 0:
        break
depth = cnt + 1 if N > (1 << cnt) else cnt

# 세그먼트 트리 생성
segment_tree = [0] * ((1 << (depth+1)) - 1) # 포화 이진 트리 생성
leaf_start = (1 << depth) - 1   # 리프 노드 시작점
leaf_idx = leaf_start
# 리프 노드에 값 채우기
for _ in range(N):
    segment_tree[leaf_idx] = int(input())
    leaf_idx += 1
# 트리 채우기
make_tree(segment_tree)

for _ in range(M+K):
    a, b, c = map(int, input().split())
    b -= 1
    # 변경
    if a == CHANGE:
        change(segment_tree, leaf_start+b, c)
    # 구간 합 계산
    elif a == SUMMATION:
        c -= 1
        print(summation(segment_tree, b, c, 0, 0, (1 << depth) - 1))
```

▼ 해설(https://www.acmicpc.net/source/37918714, 리프 → 루트 방식)

```
# CP template Version 1.006
import os
import sys
#import string
#from functools import cmp_to_key, reduce, partial
#import itertools
#from itertools import product
#import collections
#from collections import deque
#from collections import Counter, defaultdict as dd
#import math
#from math import log, log2, ceil, floor, gcd, sqrt
#from heapq import heappush, heappop
#import bisect
#from bisect import bisect_left as bl, bisect_right as br
DEBUG = False


def main(f=None):
    init(f)
    # sys.setrecursionlimit(10**9)
    # ######## INPUT AREA BEGIN ##########

    def ssum(l, r):
        res = 0
        l += 2**h - 1
        r += 2**h - 1
        while l <= r:
            if l % 2:
                res += tree[l]
                l += 1
            if not r % 2:
                res += tree[r]
                r -= 1
            l //= 2
            r //= 2
        return res

    def sudt(x, d):
        x += 2**h - 1
        tree[x] = d
        while x > 1:
            x //= 2
            tree[x] = tree[x*2] + tree[x*2+1]

    n, m, k = map(int, input().split())
    h = len(bin(n-1))-2
    tree = [0] * 2**(h+1)

    for i in range(n):
        tree[2**h+i] = int(input().strip())

    for i in range(h-1, -1, -1):
        for j in range(2**i, 2**(i+1)):
            tree[j] = tree[j*2] + tree[j*2+1]

    for _ in range(m+k):
```

```
        a, b, c = map(int, input().split())
        if a == 1:
            sudt(b, c)
        else:
            print(ssum(b, c))


    # ######## INPUT AREA END ############


# TEMPLATE ################################


enu = enumerate


def For(*args):
    return itertools.product(*map(range, args))


def Mat(h, w, default=None):
    return [[default for _ in range(w)] for _ in range(h)]


def nDim(*args, default=None):
    if len(args) == 1:
        return [default for _ in range(args[0])]
    else:
        return [nDim(*args[1:], default=default) for _ in range(args[0])]


def setStdin(f):
    global DEBUG, input
    DEBUG = True
    sys.stdin = open(f)
    input = sys.stdin.readline


def init(f=None):
    global input
    input = sys.stdin.readline  # io.BytesIO(os.read(0, os.fstat(0).st_size)).readline
    if os.path.exists("o"):
        sys.stdout = open("o", "w")
    if f is not None:
        setStdin(f)
    else:
        if len(sys.argv) == 1:
            if os.path.isfile("in/i"):
                setStdin("in/i")
            elif os.path.isfile("i"):
                setStdin("i")
        elif len(sys.argv) == 2:
            setStdin(sys.argv[1])
        else:
            assert False, "Too many sys.argv: %d" % len(sys.argv)


def pr(*args):
    if DEBUG:
```

```
        print(*args)


def pfast(*args, end="\n", sep=' '):
    sys.stdout.write(sep.join(map(str, args)) + end)


def parr(arr):
    for i in arr:
        print(i)


if __name__ == "__main__":
    main()
```

참고 자료

- 구간 합: https://hroad.tistory.com/52
- 세그먼트 트리: https://m.blog.naver.com/PostView.naver?blogId=kks227&logNo=220791986409&navType=by
- 세그먼트 트리 파이썬 코드: https://velog.io/@corone_hi/구간-트리-Segment-Tree-Python