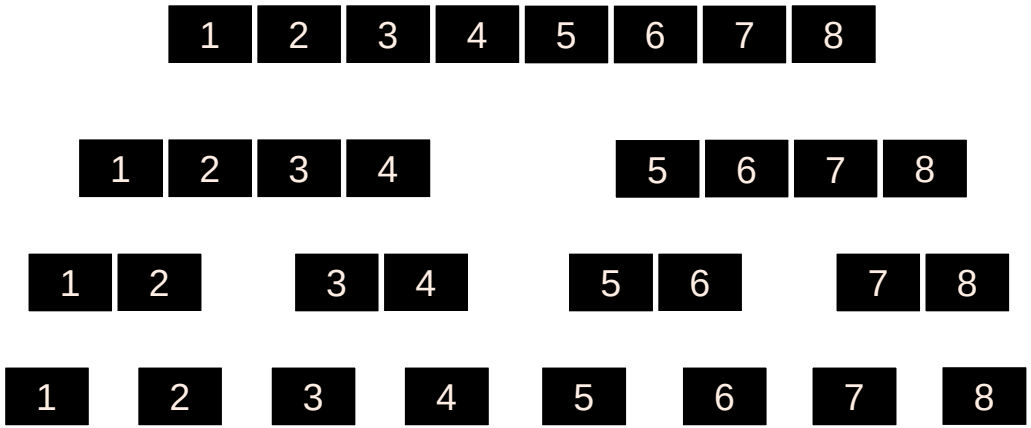
Segment Tree
중급 5회차

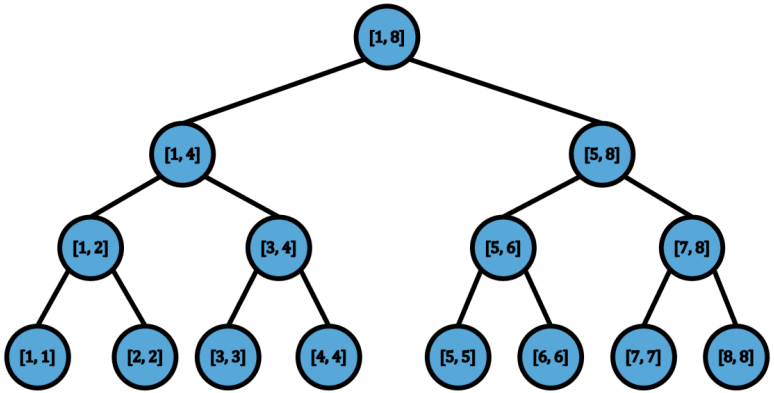
서강대학교 전해성(seastar105)

배열에서 구간의 정보를 저장한다.

세그먼트 트리는 배열에서 구간에 대한 정보를 처리할 때 주로 사용됩니다.

아래처럼 전체 배열을 반으로 쪼개는 과정을 살펴보면 이진 트리와 같이 생겼습니다.

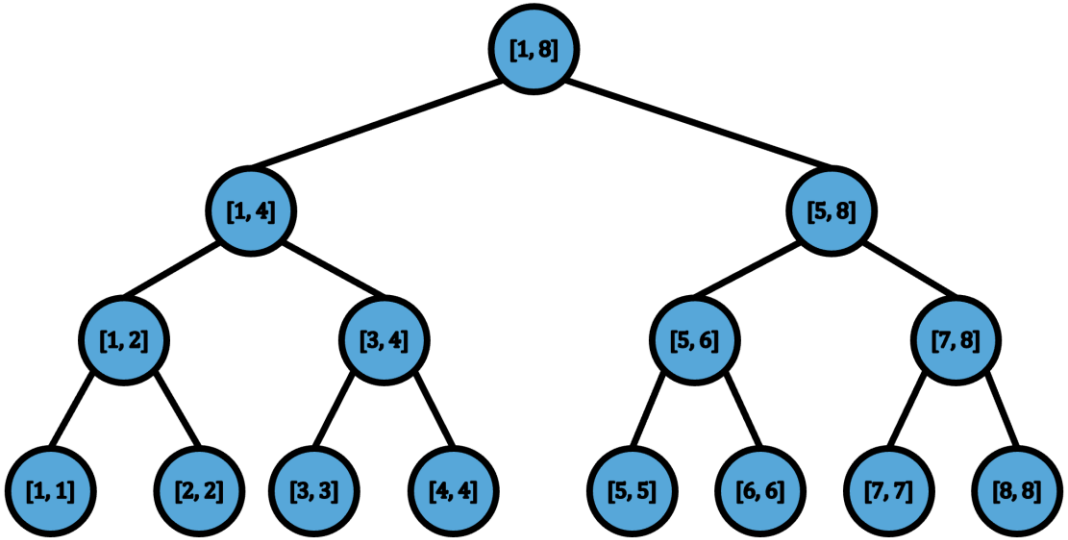




배열에서 구간의 정보를 저장한다.

아래와 같은 트리에서 노드 별로 담당 구간에 대한 정보를 가지고 있습니다.

이 정보는 구간의 합, 최솟값, 최댓값이 될 수도 있고 해당 구간의 원소들을 정렬한 배열일 수도 있고 gcd일 수도 있고 정말 다양하게 있을 수 있습니다.



백준 2042번 구간 합 구하기

길이 N짜리 배열이 주어지고 두 종류의 쿼리를 처리해야 합니다.

첫 번째 쿼리는 특정 위치의 원소를 바꾸는 것입니다.

두 번째 쿼리는 특정 구간의 합을 구하는 것입니다.

배열로만 문제를 해결하면 첫 번째 쿼리는 O(1), 두 번째 쿼리는 O(N)만큼 걸립니다.

백준 2042번 구간 합 구하기

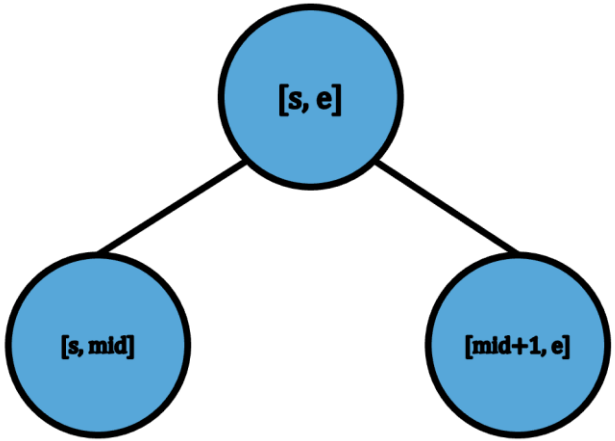
세그먼트 트리를 사용하면 1, 2번 쿼리를 O(lg N)에 해결이 가능합니다.

먼저 배열이 주어졌을 때 노드가 자신이 맡고 있는 구간의 합을 저장하고 있는 세그먼트 트리를 만들어 봅시다.

세그먼트 트리 구성

세그먼트 트리는 이진 트리입니다.

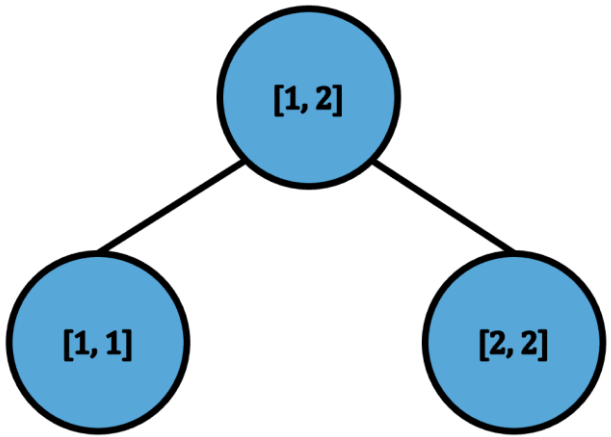
어떤 노드가 담당하고 있는 구간이 [s, e]이고 그 길이가 1보다 크다면 구간을 반으로 쪼개서 각각 한쪽은 왼쪽 자식 노드, 한쪽은 오른쪽 자식노드가 담당하도록 합니다.



세그먼트 트리 구성

이런 식으로 쪼개는 과정을 반복하다 보면 구간의 길이가 1인 노드들이 생깁니다.

이런 노드들은 필연적으로 리프노드가 될 것입니다. 리프 노드의 구간이 [i,i]라면 가져야 할 값은 배열의 i번째 값입니다.



세그먼트 트리 구성

이제 재귀적으로 트리를 만들어 봅시다.

노드가 맡은 구간의 길이가 1일 때는 리프 노드로 base case입니다.

리프노드가 아닌 노드들은 왼쪽 자식과 오른쪽 자식의 합을 더한 것을 자신의 합으로 갖고 있습니다.

```
def build_segtree(cur_node, node_l, node_r):
    if node_l == node_r:
        cur_node.sum = a[node_l]
        return
    mid = (node_l + node_r) / 2
    cur_node.left = build(left_child_node, node_l, mid)
    cur_node.right = build(right_child_node, mid+1, node_r)
    cur_node.sum = cur_node.left.sum + cur_node.right.sum
```

세그먼트 트리 구성 - 구현

우리가 만들 트리는 이진트리이고 배열의 길이를 알고 있기 때문에 트리를 만들 때 배열을 사용할 수 있습니다.

트리 배열에서 루트 노드의 인덱스를 1이라고 하면 모든 노드의 왼쪽 자식은 2*i, 오른쪽 자식은 2*i+1로 나타낼 수 있습니다.

수열의 길이가 N이라면 전체 트리 배열의 길이는 $2^{\lceil \lg N \rceil + 1}$ 로 잡으면 충분합니다.

```
ll tree[1<<21];
int a[1000005];
void build_segtree(int node_idx, int node_left, int node_right) {
    if (node_left == node_right) {
        tree[node_idx] = a[node_left];
        return ;
    }
    int mid = (node_left + node_right) / 2;
    build_segtree(node_idx * 2, node_left, mid);
    build_segtree(node_idx * 2 + 1, mid + 1, node_right);
    tree[node_idx] = tree[node_idx * 2] + tree[node_idx * 2 + 1];
}
```

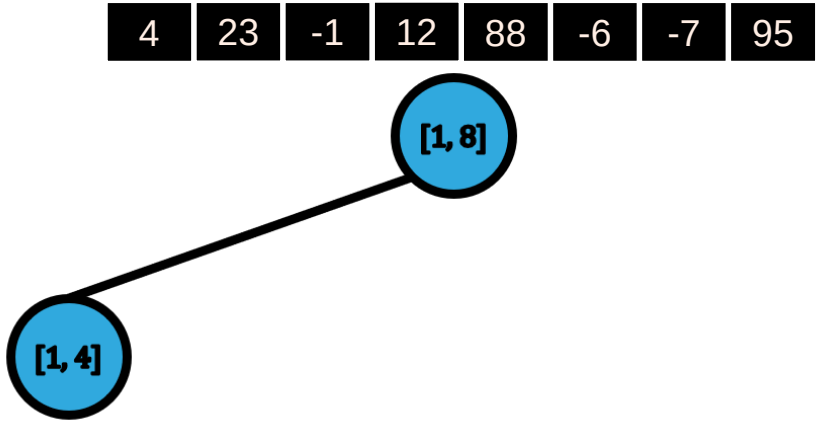
세그먼트 트리 구성 - 예시

4	23	-1	12	88	-6	-7	95

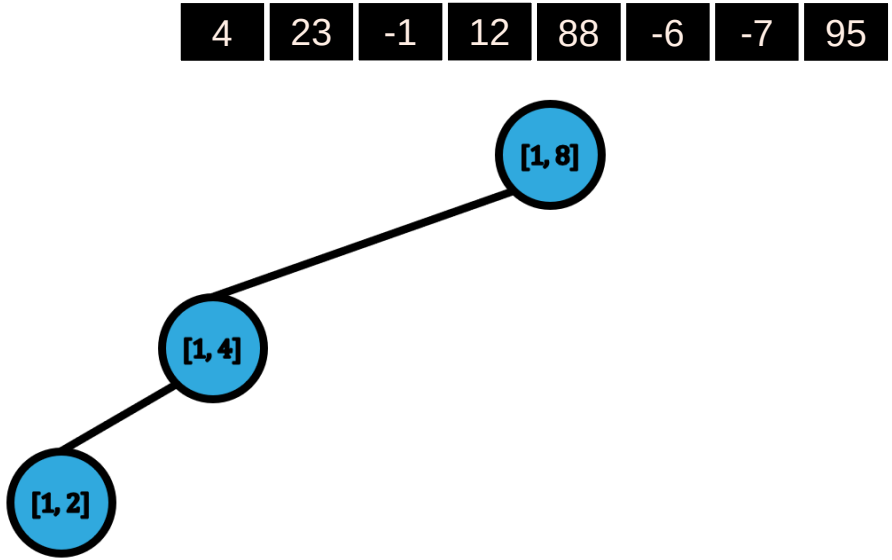
위와 같은 길이 8짜리 배열에 대해서 세그먼트 트리를 만들어봅시다.

build_tree(1, 1, 8)을 호출해서 루트 노드를 만드는 것을 시작으로 재귀적으로 트리를 만들 수 있습니다.

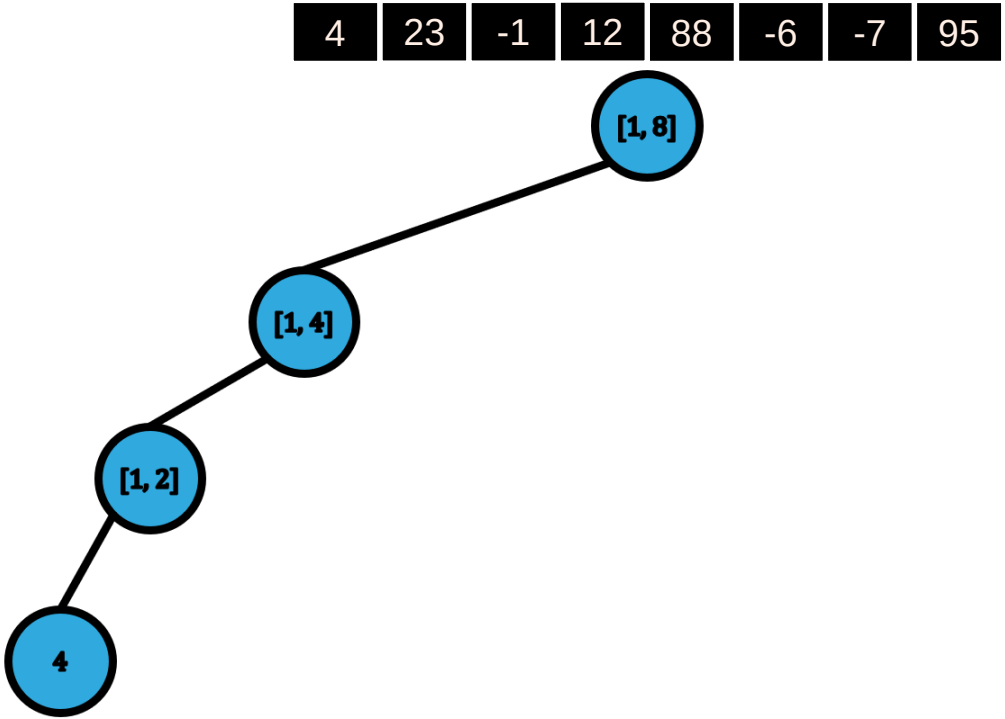
세그먼트 트리 구성 - 예시



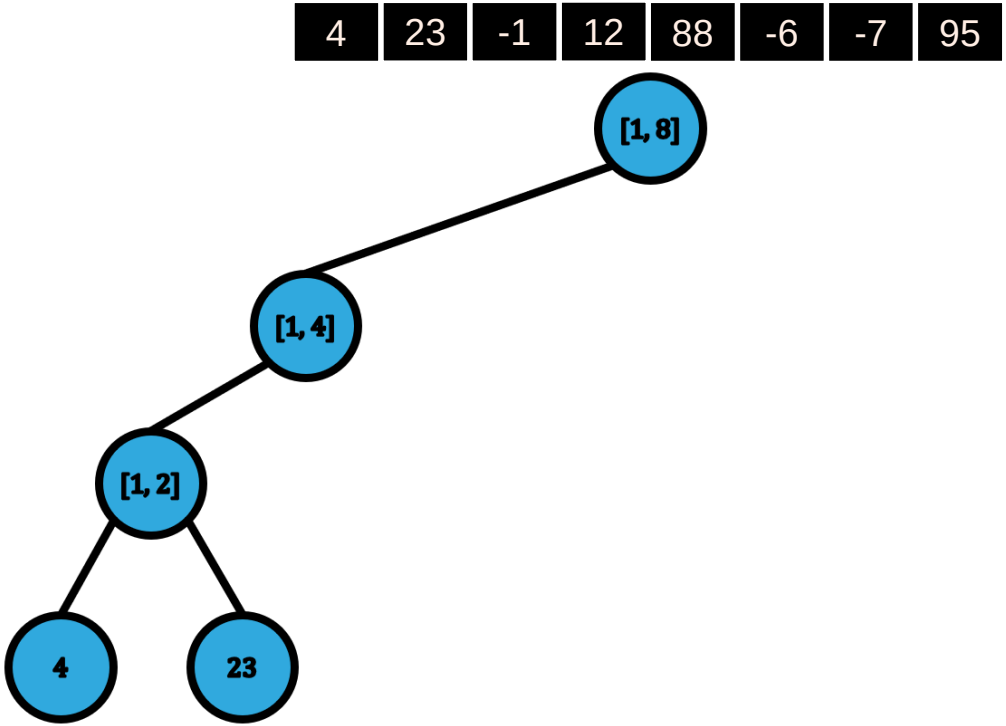
세그먼트 트리 구성 - 예시



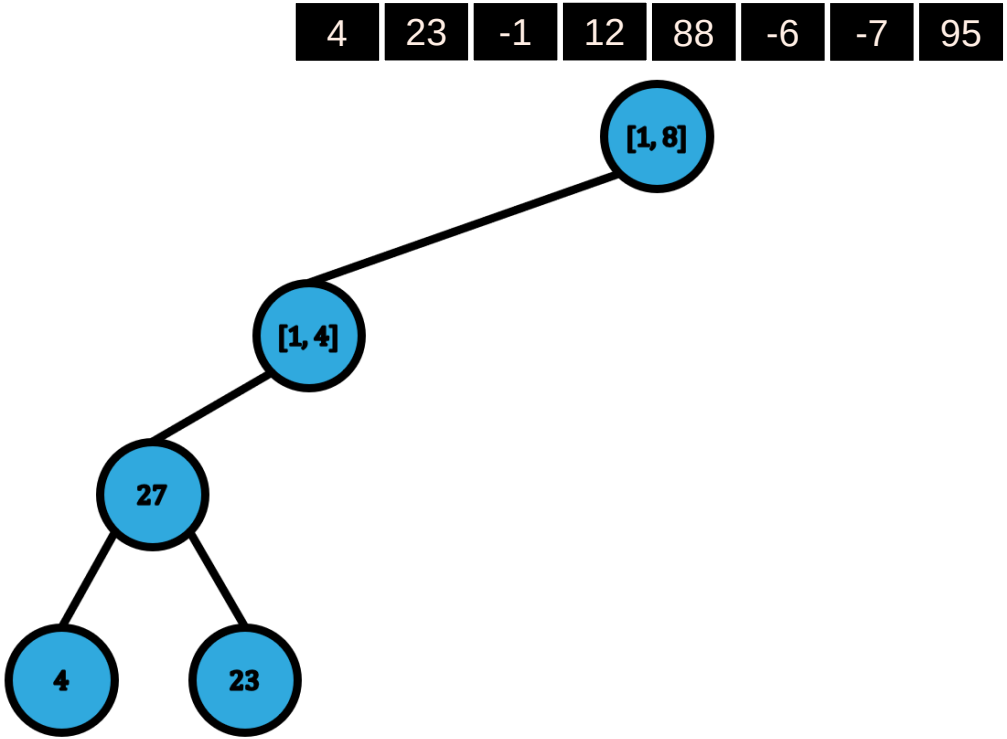
세그먼트 트리 구성 - 예시



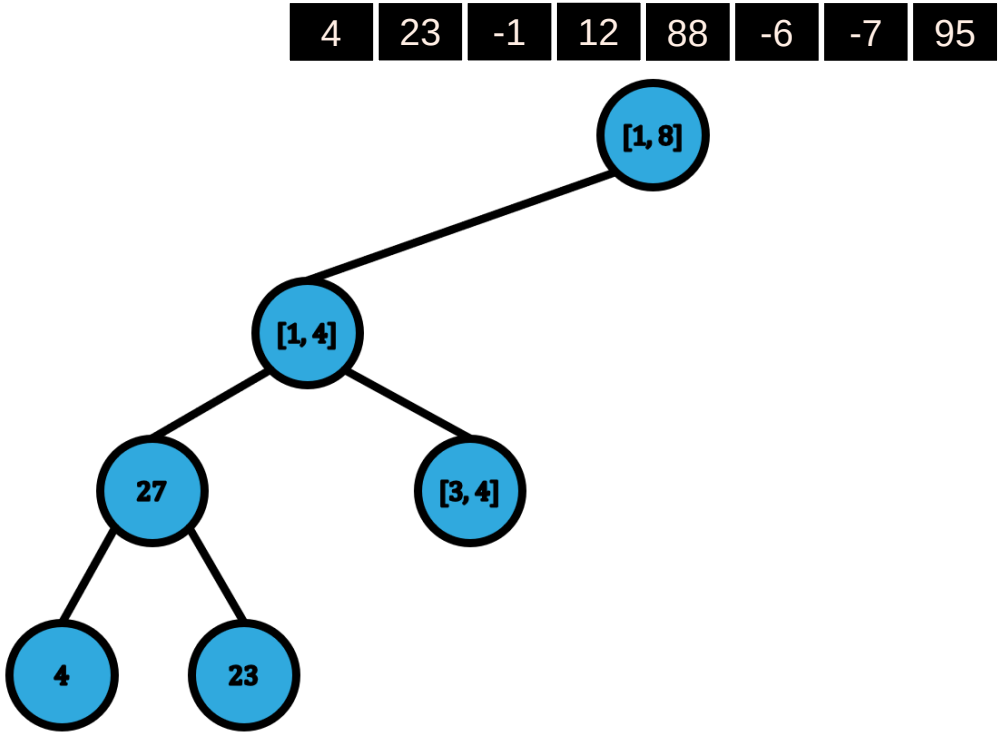
세그먼트 트리 구성 - 예시



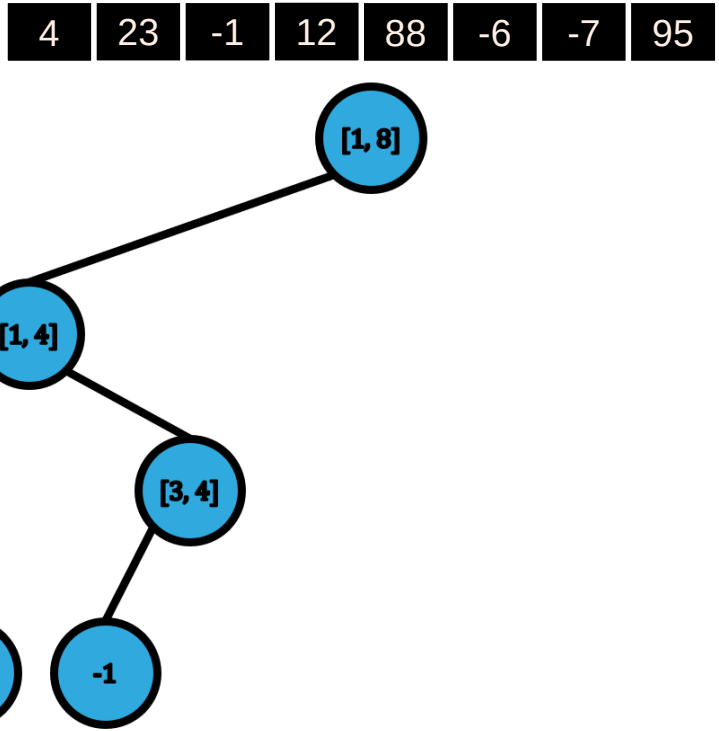
세그먼트 트리 구성 - 예시



세그먼트 트리 구성 - 예시

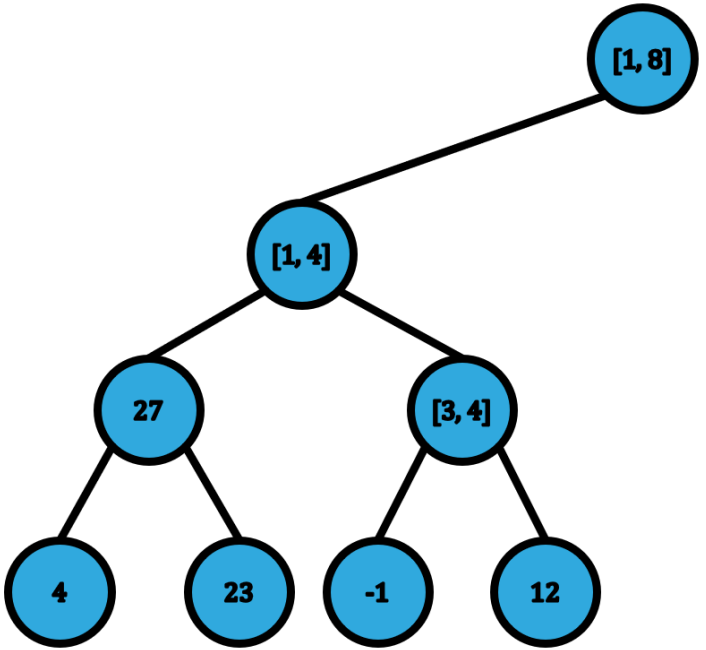


세그먼트 트리 구성 - 예시

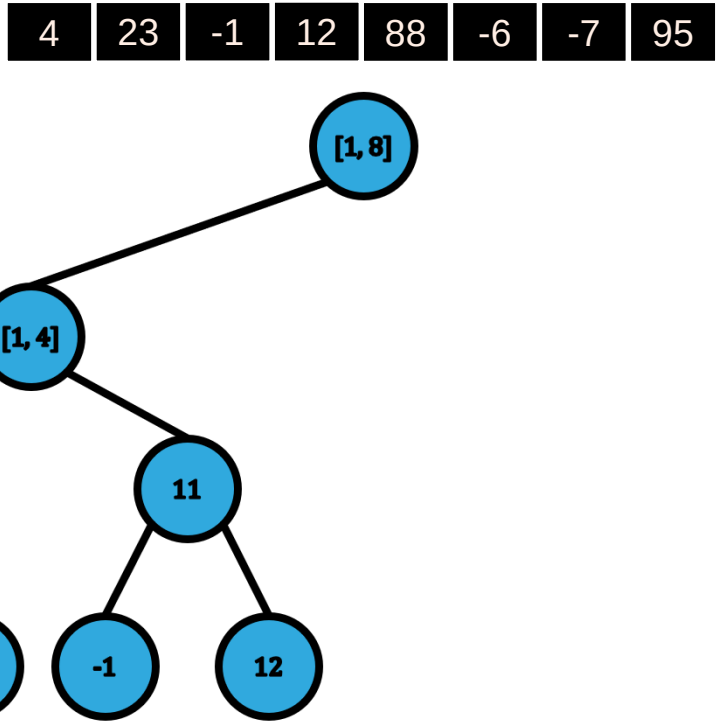


세그먼트 트리 구성 - 예시

4	23	-1	12	88	-6	-7	95

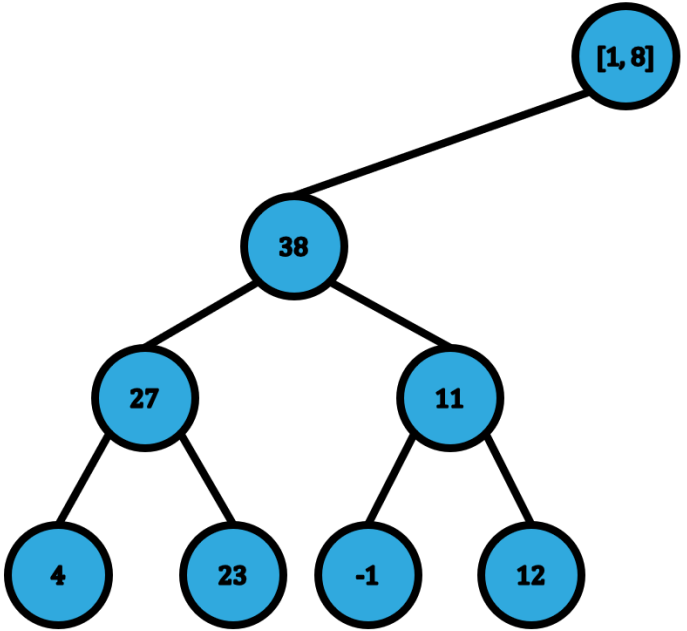


세그먼트 트리 구성 - 예시

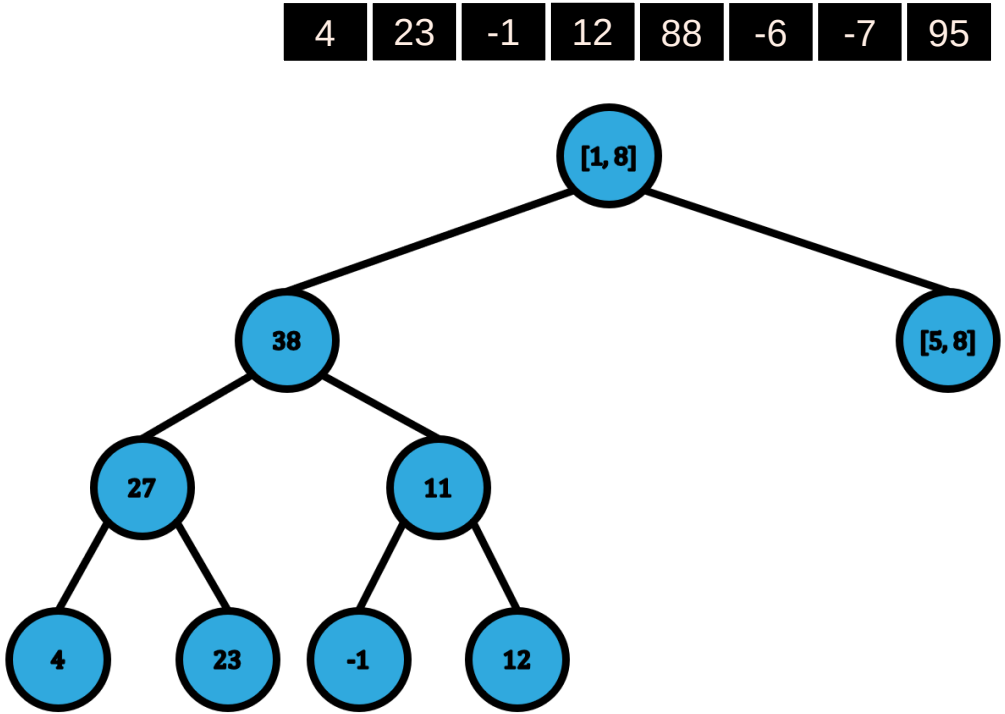


세그먼트 트리 구성 - 예시

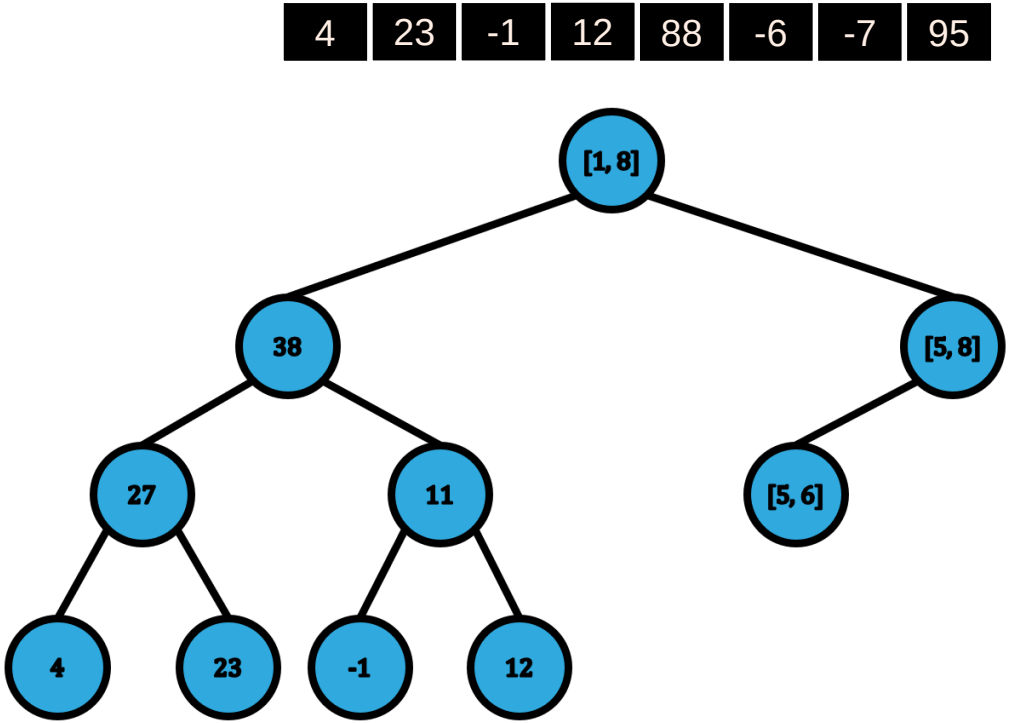
4	23	-1	12	88	-6	-7	95



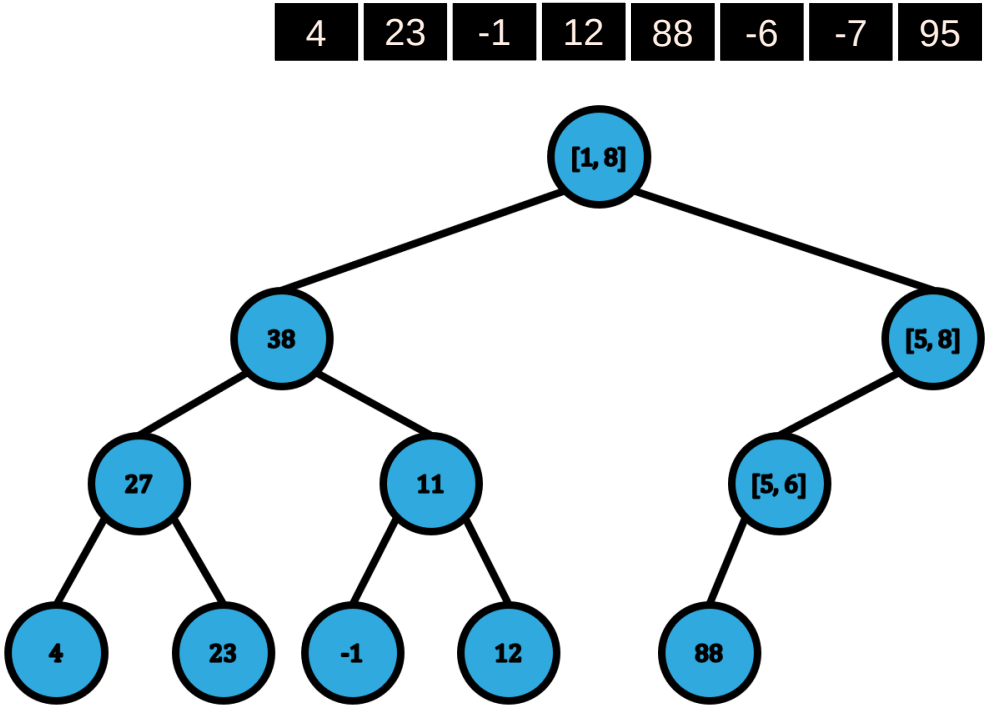
세그먼트 트리 구성 - 예시



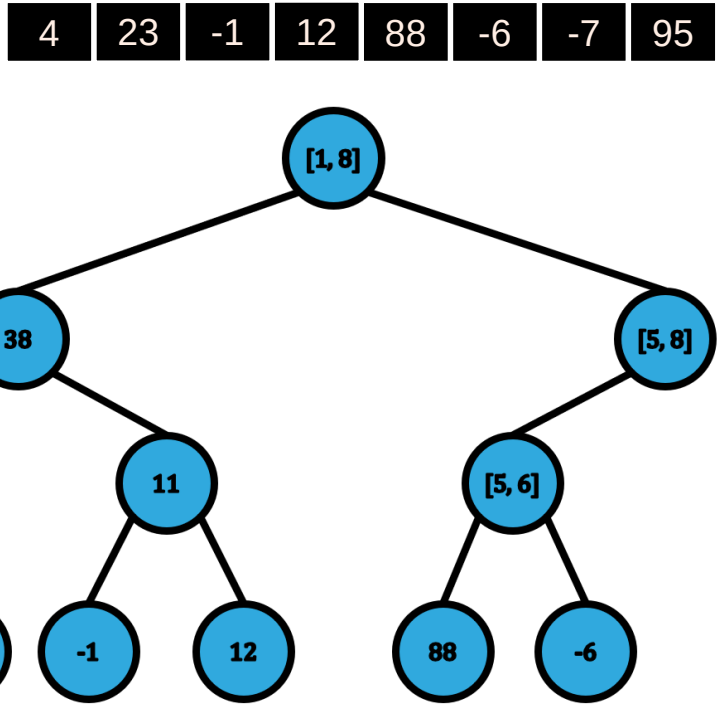
세그먼트 트리 구성 - 예시



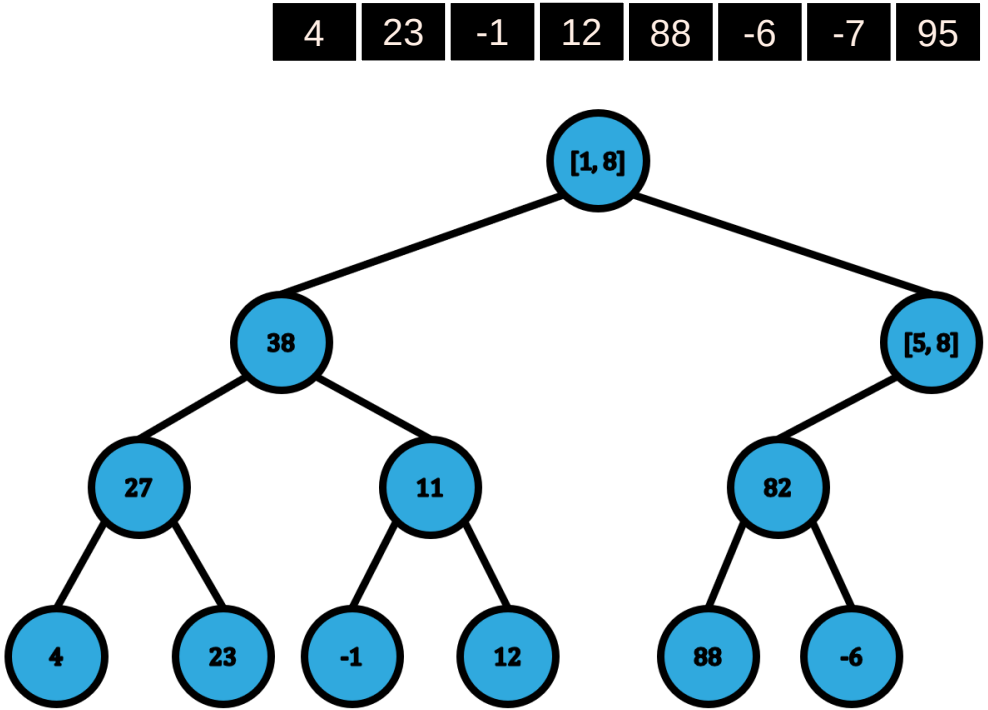
세그먼트 트리 구성 - 예시



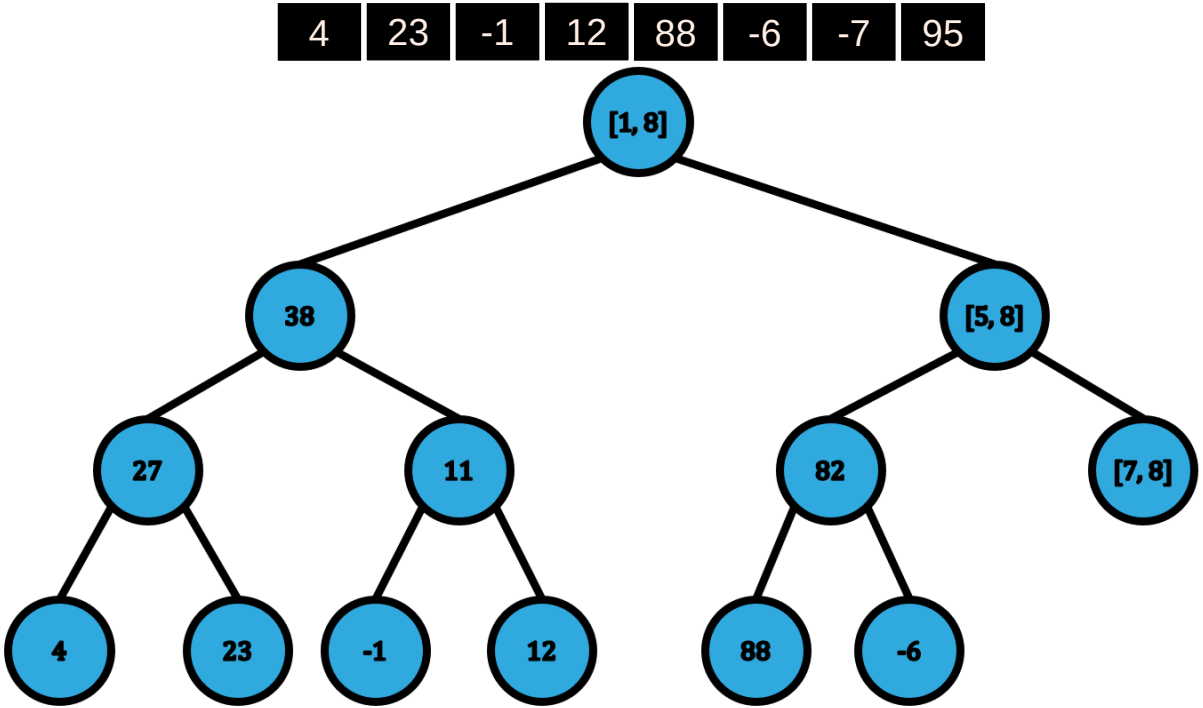
세그먼트 트리 구성 - 예시



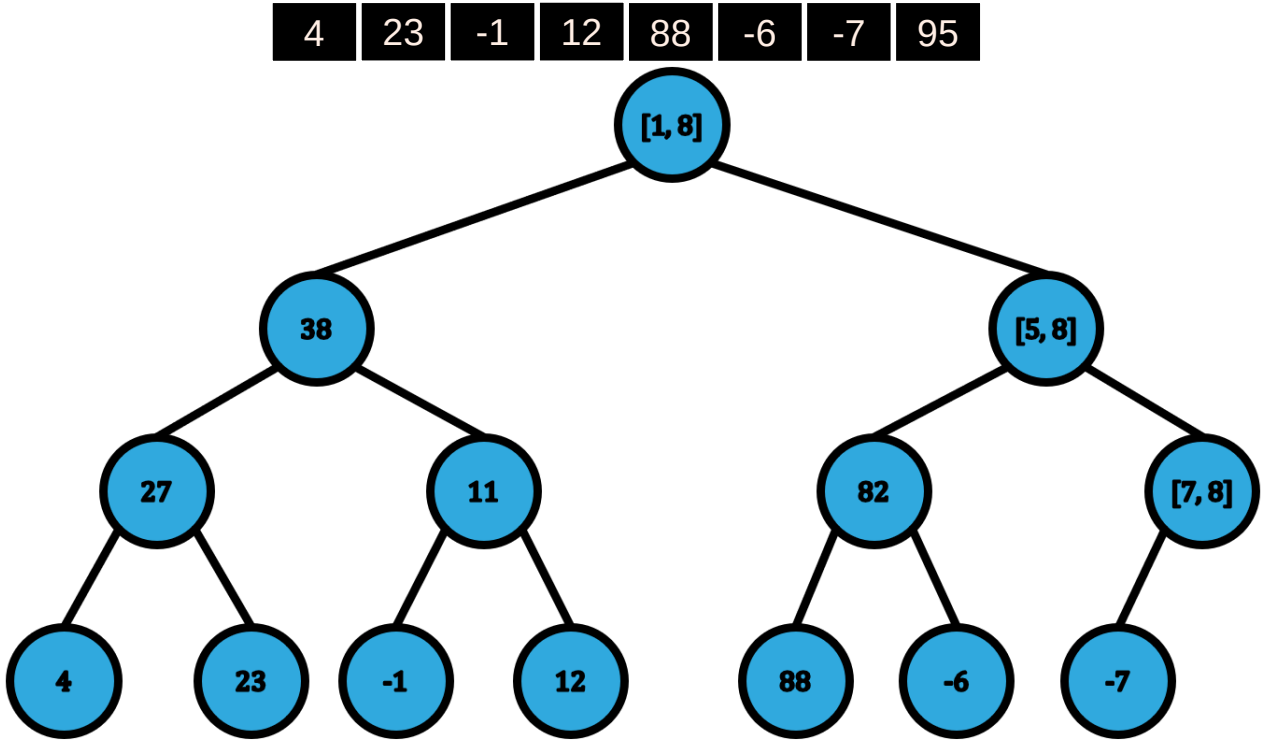
세그먼트 트리 구성 - 예시



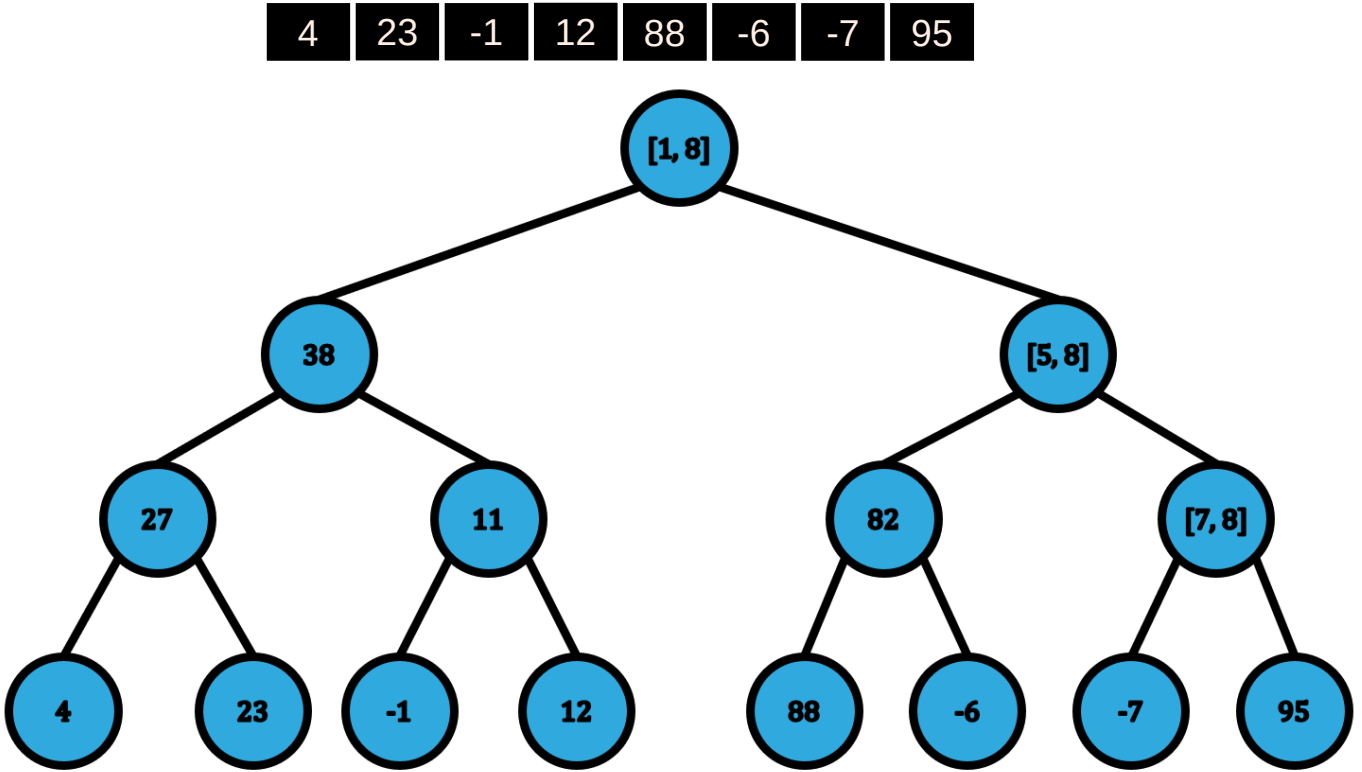
세그먼트 트리 구성 - 예시



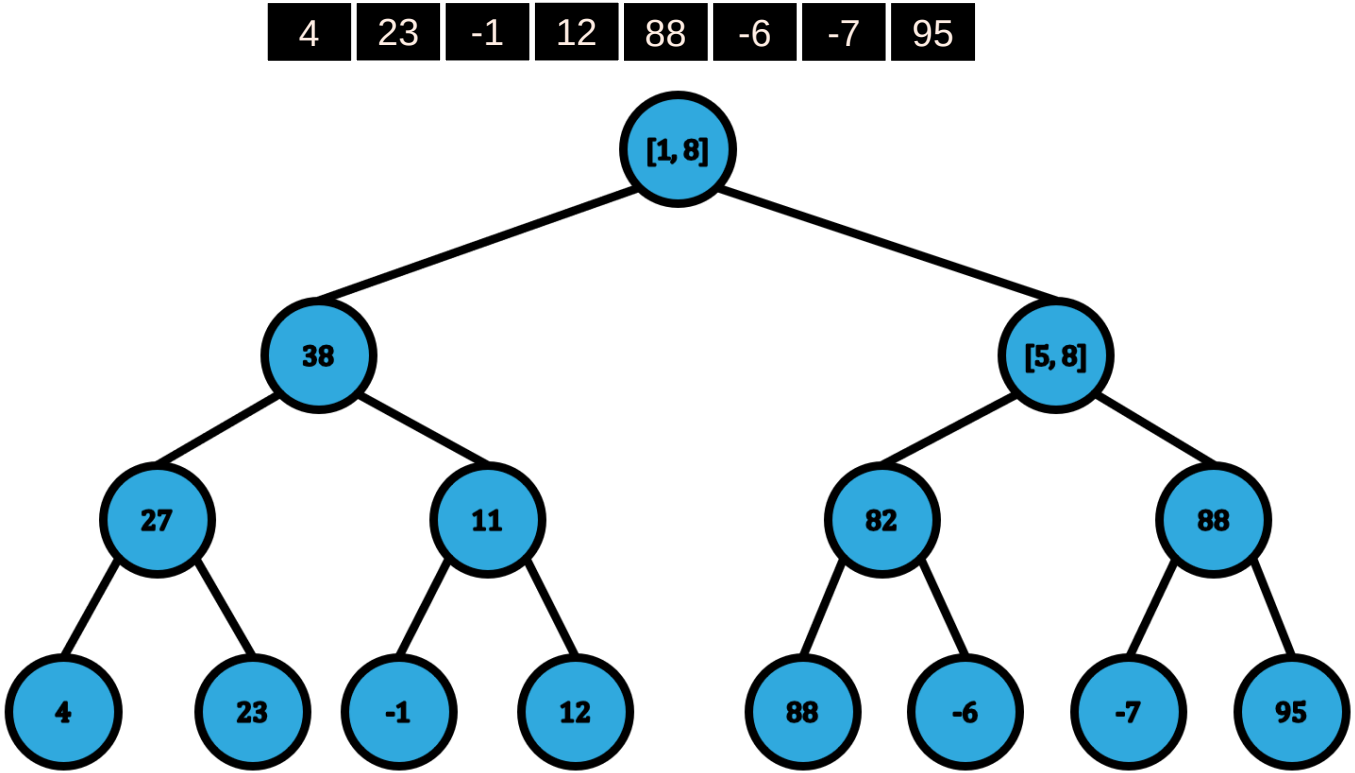
세그먼트 트리 구성 - 예시



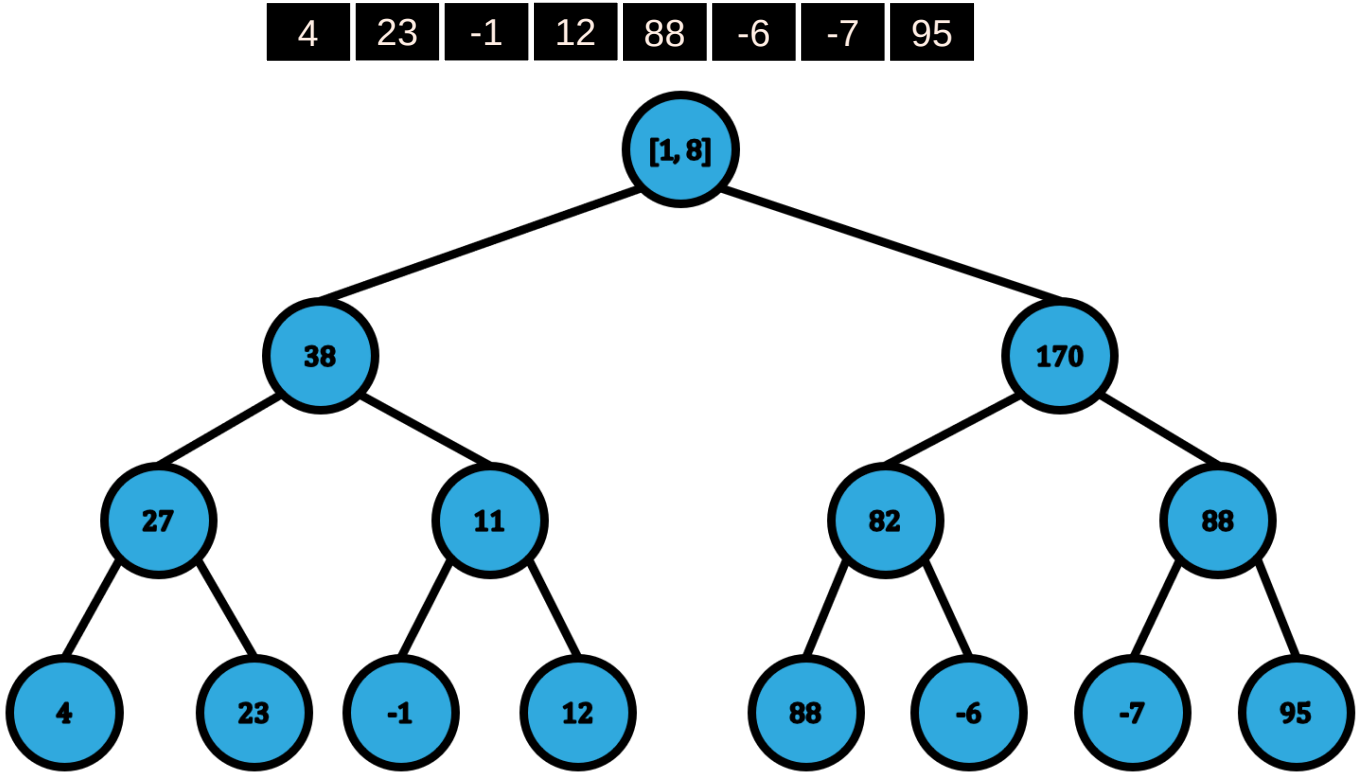
세그먼트 트리 구성 - 예시



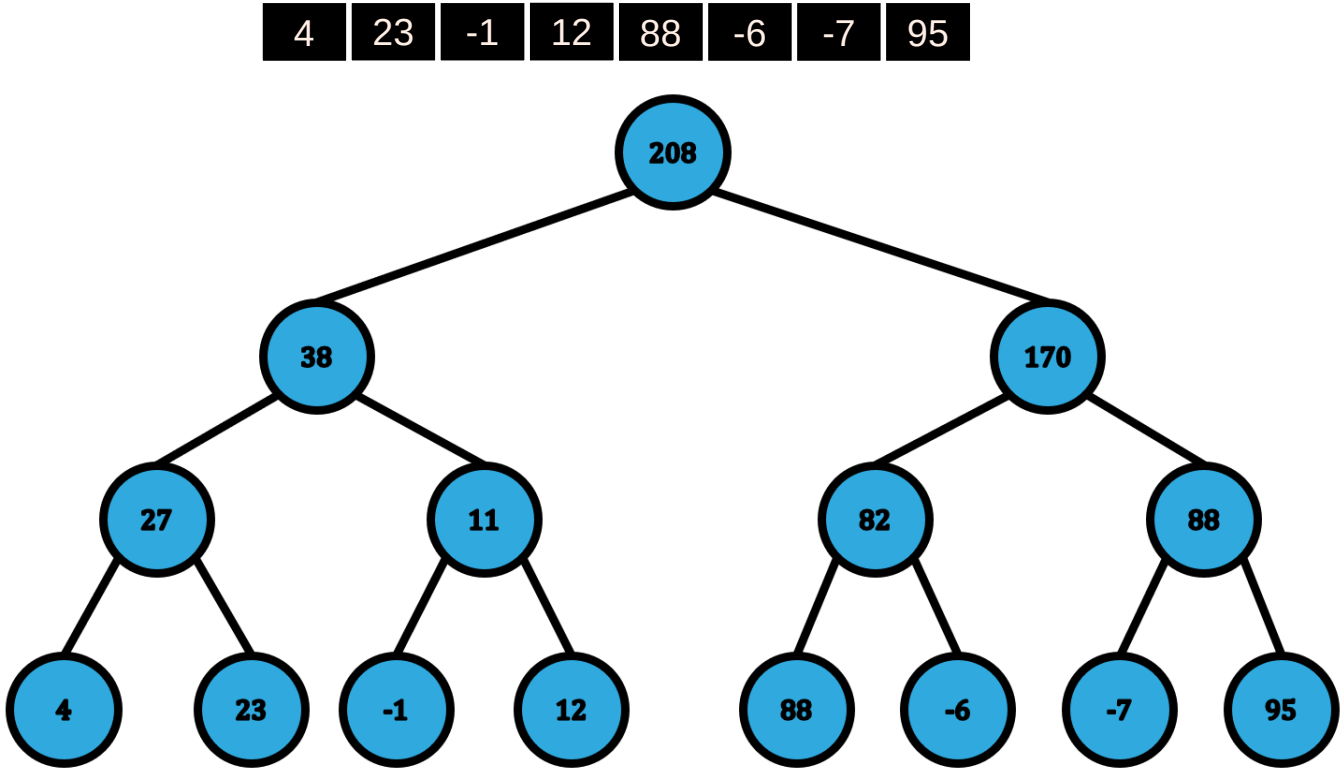
세그먼트 트리 구성 - 예시



세그먼트 트리 구성 - 예시



세그먼트 트리 구성 - 예시



세그먼트 트리 구성 - 시간복잡도

이와 같은 방식으로 세그먼트 트리를 만들었을 때의 시간복잡도는 O(N)입니다.

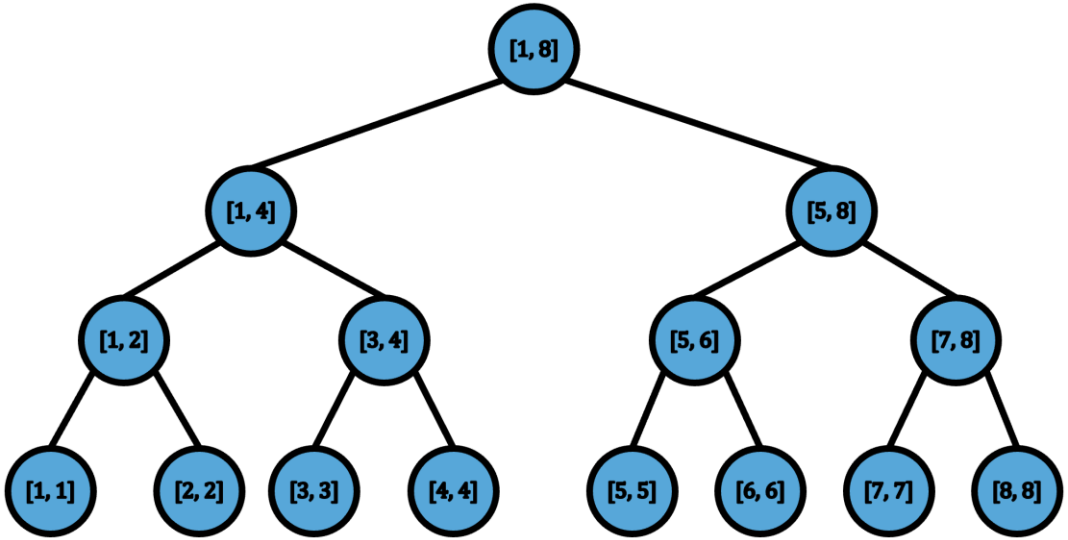
각 노드 별로 한 번씩 방문하게 되고 자식 노드 두개의 합을 합치는데는 O(1)의 연산이 필요합니다.

총 노드의 수는 O(N)이기 때문에 시간복잡도 또한 그렇습니다.

세그먼트 트리 구간 쿼리

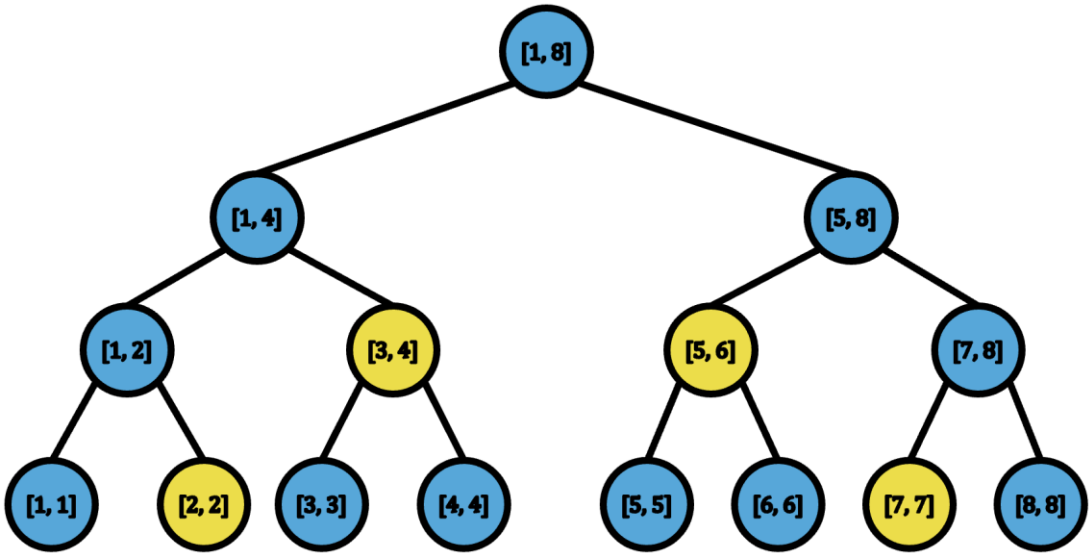
세그먼트 트리로 두 번째 쿼리를 처리하는 방법부터 살펴봅시다.

일단 아래와 같이 트리를 구성했고 각 노드는 자신이 맡은 구간의 합을 들고 있다고 합시다.



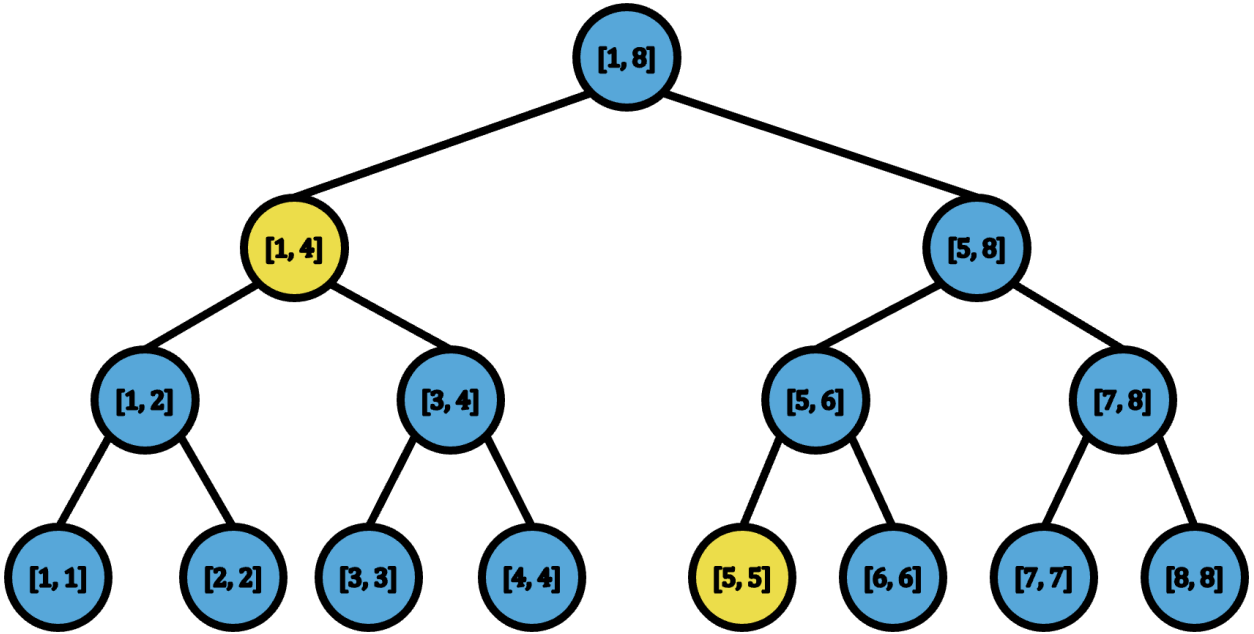
세그먼트 트리 구간 쿼리

[2, 7]의 구간 합을 구하는 쿼리가 들어오면 우리가 알아야 되는 노드들은 아래와 같습니다.



세그먼트 트리 구간 쿼리

쿼리가 [1, 5]일 땐 다음과 같습니다.



세그먼트 트리 구간 쿼리

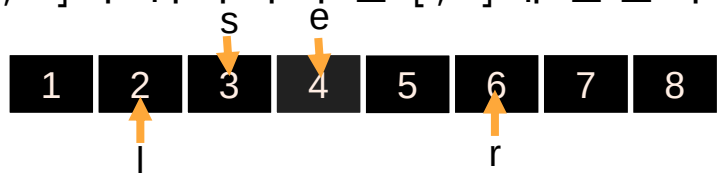
일반화를 해서 쿼리로 들어온 구간이 [l, r]이라고 합시다.

그리고 각 노드들이 들고 있는 구간을 [s, e]라고 합시다.

노드가 담당하고 있는 구간에 따라서 할 일은 달라집니다.

세그먼트 트리 구간 쿼리

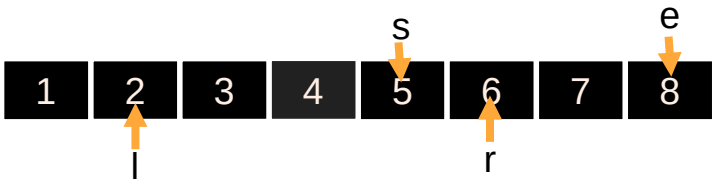
Case 1) 노드의 구간 [s, e]가 쿼리의 구간 [l, r]에 완전히 포함될 때



Case 2) 노드의 구간 [s, e]가 쿼리의 구간 [l, r]과 겹치는 부분이 없을 때

세그먼트 트리 구간 쿼리

Case 3) 노드의 구간 [s, e]가 쿼리의 구간 [l, r]에 포함되지 않고 겹칠 때

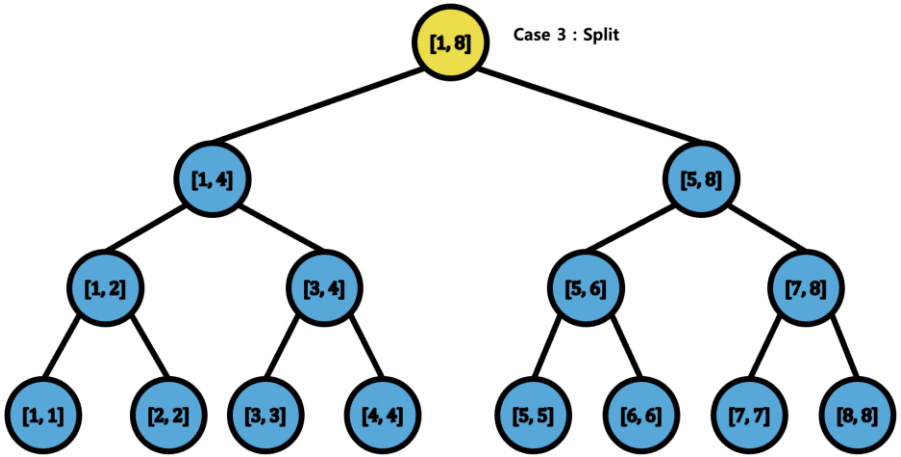


이럴 경우에는 [s, e]를 반으로 쪼개고 [s, mid]와 [mid+1, e]에 대해서 위 과정을 수행합니다.

세그먼트 트리 구간 쿼리

쿼리로 [2, 7]이 들어왔을 때 위 과정이 어떻게 되는지 봅시다.

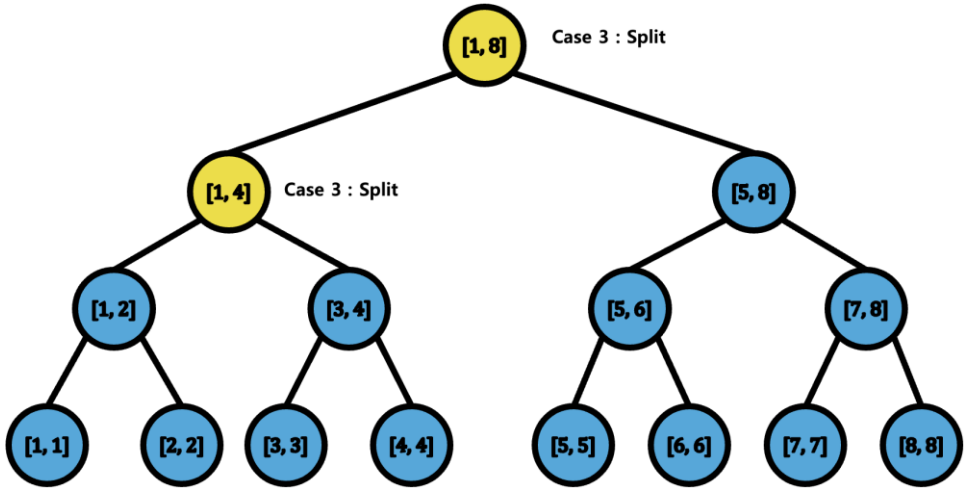
루트 노드는 [1, 8]의 정보를 가지고 있으며 쿼리 구간인 [2, 7]에 포함되지 않으면서 겹칩니다.
[1, 4]와 [5, 8]로 구간을 나눠서 [2, 7]의 구간 합을 구하도록 합니다.



세그먼트 트리 구간 쿼리

세그먼트 트리 상에서 현재 노드의 구간 [s, e]를 반으로 나눠서 본다는 것은 자식 노드의 정보를 사용하는 것과 같습니다.

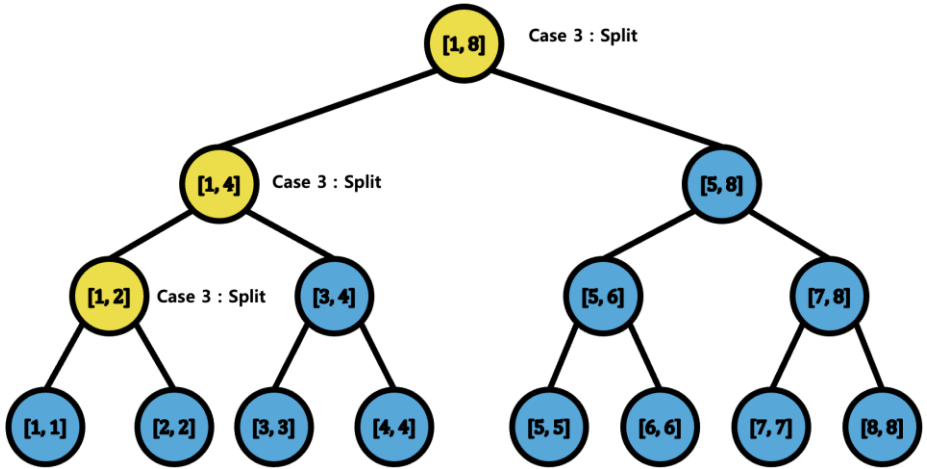
왼쪽 자식 노드가 [1, 4]의 정보를 담고 있습니다. [1, 4] 또한 [2, 7]에 포함되지 않으면서 겹칩니다. 구간을 쪼갭시다.



세그먼트 트리 구간 쿼리

왼쪽 자식 노드의 구간인 [1,2]를 살펴봅시다. [2, 7]과 다시 또 포함되진 않지만 겹칩니다.

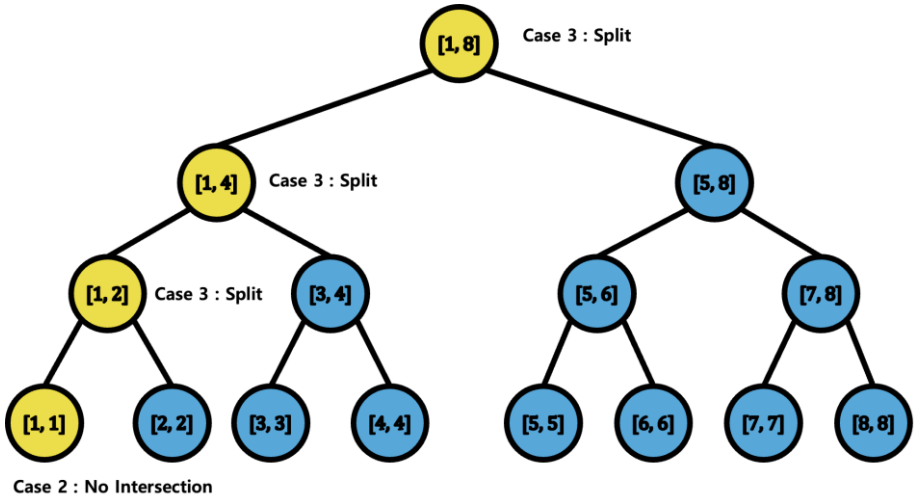
[1,1]과 [2,2]로 쪼개야 하니 왼쪽 자식 노드로 내려갑니다.



세그먼트 트리 구간 쿼리

노드의 구간 [1, 1]과 쿼리의 구간 [2, 7]은 겹치는 부분이 전혀 없습니다.

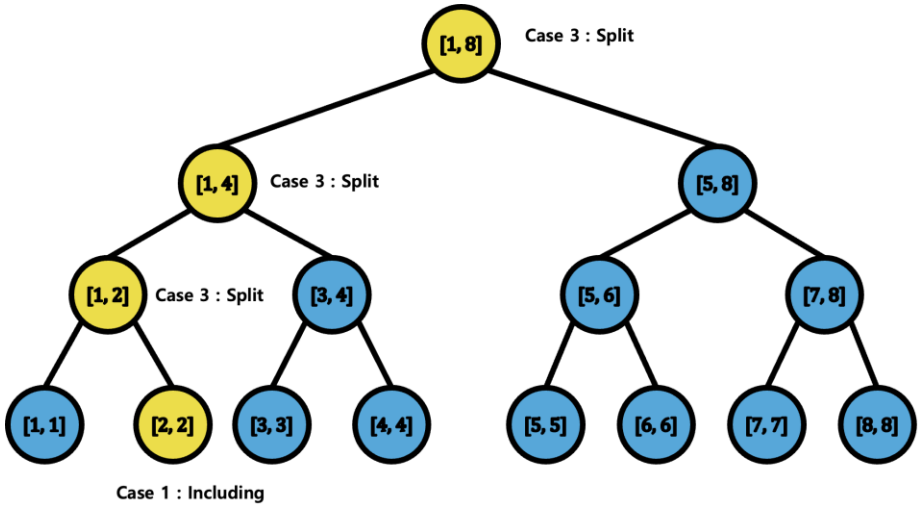
이 구간에는 쿼리에서 원하는 값이 없습니다.



세그먼트 트리 구간 쿼리

[1, 1]에서 [1, 2]의 노드로 돌아와서 [2, 2]로 진행합니다.

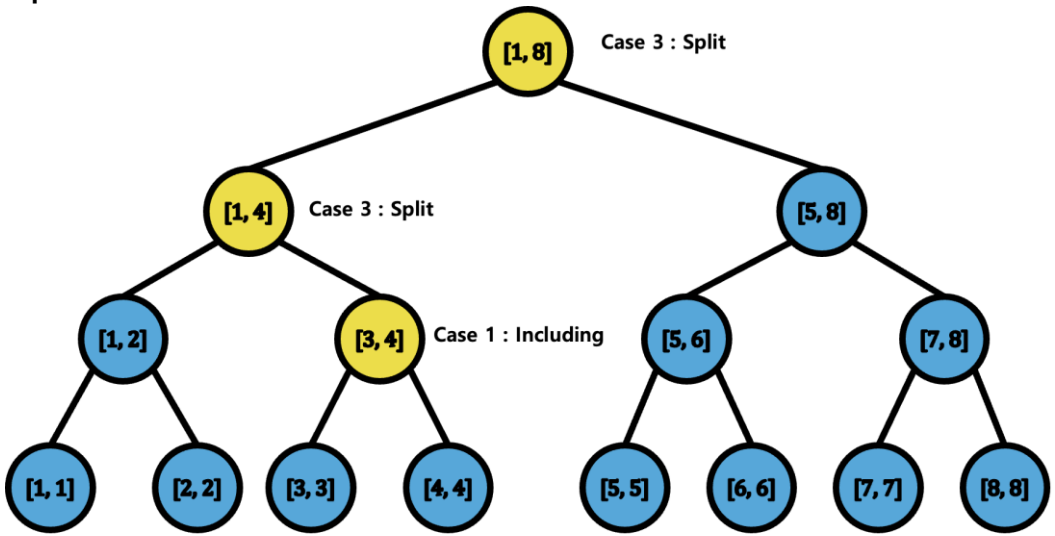
[2, 2]는 쿼리 구간 [2, 7]에 완전히 포함됩니다. 답에 반영하고 끝납니다.



세그먼트 트리 구간 쿼리

[2, 2]에서 [1, 2]로, [1, 2]에서 [1, 4]로 돌아온 뒤엔 [3, 4]의 노드를 살펴봅니다.

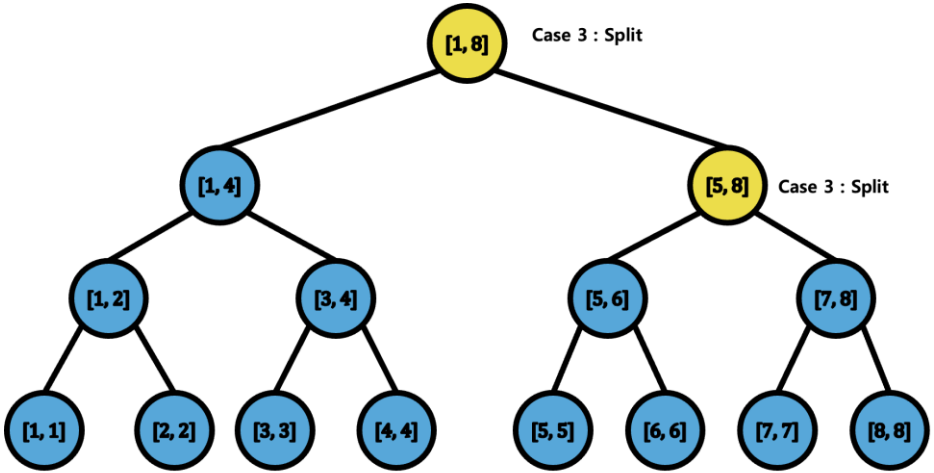
[3, 4]도 쿼리 구간 [2, 7]에 완전히 포함되기 때문에 답에 이 노드의 정보를 반영하고 끝납니다.



세그먼트 트리 구간 쿼리

[3, 4]의 노드에서 [1, 4]의 노드로, [1, 4]의 노드에서 [1, 8]의 노드로 돌아오면 다시 아직 안 본 반쪽인 [5, 8]의 노드를 살펴 봅니다.

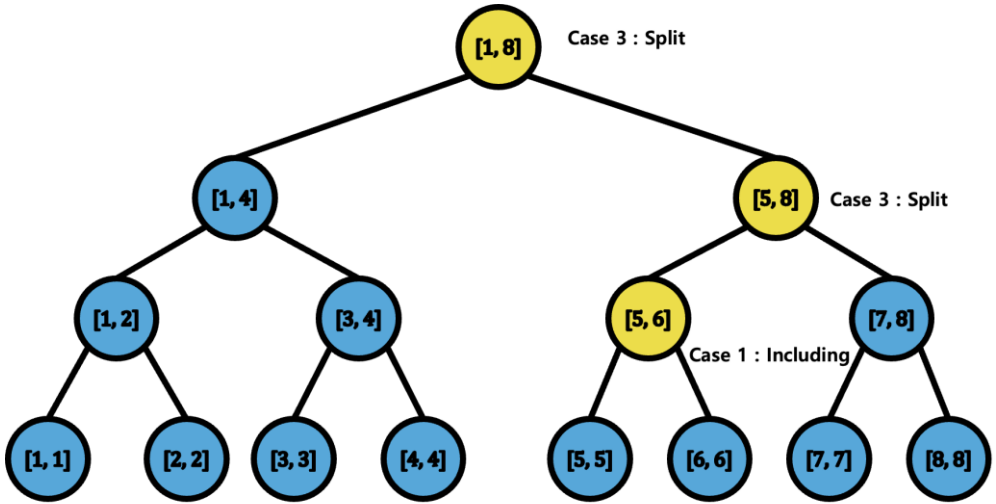
[5, 8]은 쿼리 구간에 포함되지 않으면서 겹치기 때문에 다시 구간을 쪼갭니다.



세그먼트 트리 구간 쿼리

[5, 8]의 왼쪽 절반인 [5, 6]은 [2, 7]에 완전히 포함됩니다.

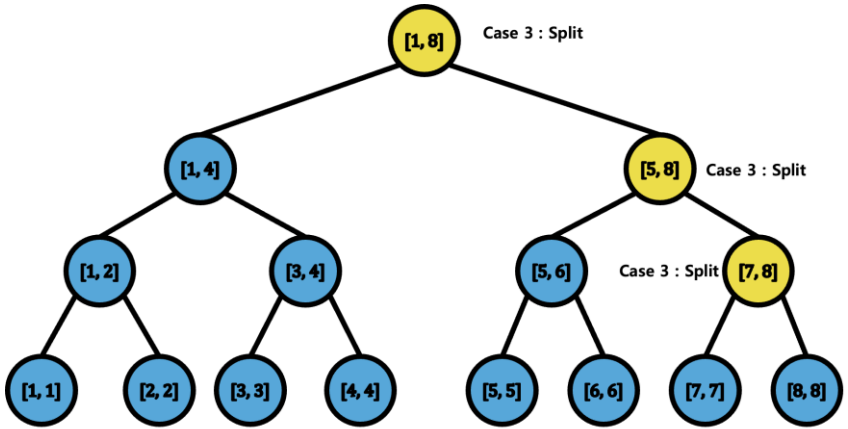
답에 반영하고 끝냅니다.



세그먼트 트리 구간 쿼리

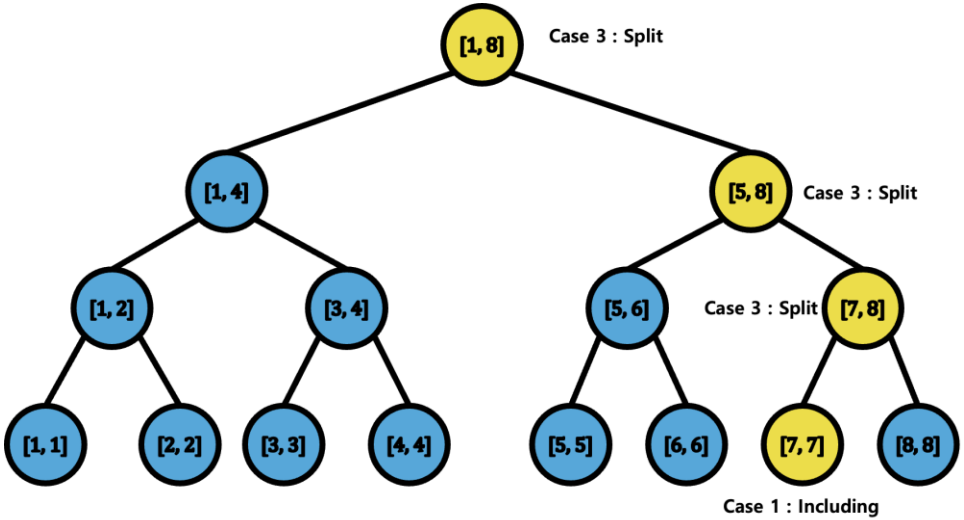
[5, 8]로 돌아온 뒤에 오른쪽 절반인 [7, 8]의 노드로 내려갑니다.

[7, 8]은 [2, 7]에 포함되지 않으면서 겹치는 부분이 있습니다. 절반으로 쪼갭니다.



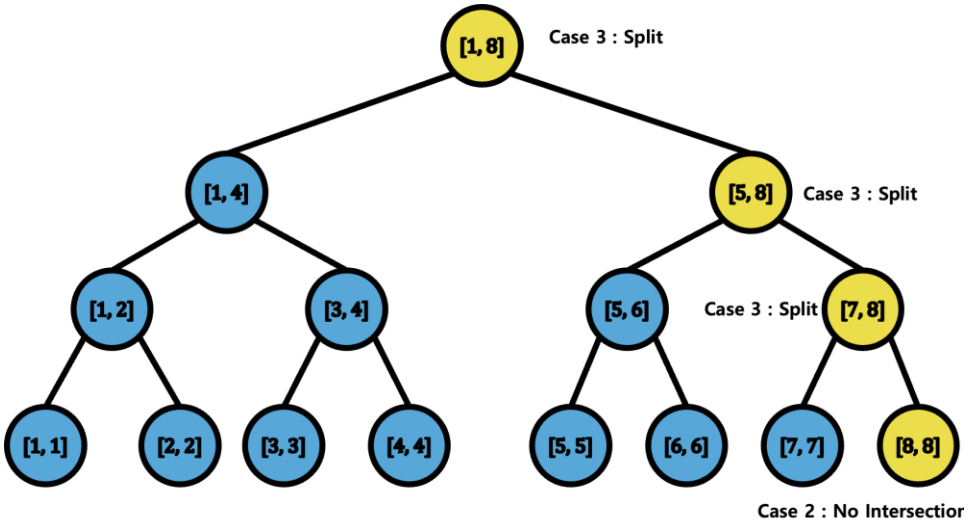
세그먼트 트리 구간 쿼리

[7, 7]은 쿼리 구간에 포함됩니다. 답에 반영하고 끝내도록 합니다.



세그먼트 트리 구간 쿼리

[8, 8]은 아예 겹치는 부분이 없습니다. 답에 반영하지 않고 끝냅니다.



세그먼트 트리 구간 쿼리 - 코드

이런 구간 쿼리 또한 재귀적으로 처리가 가능합니다.

```
ll query(int node_idx, int node_left, int node_right, int query_left, int query_right) {
    if (node_left > query_right || node_right < query_left) return 0;
    if (query_left <= node_left && node_right <= query_right) return tree[node_idx];
    int mid = (node_left + node_right) / 2;
    return query(node_idx * 2, node_left, mid, query_left, query_right) +
           query(node_idx * 2 + 1, mid + 1, node_right, query_left, query_right);
}
```

세그먼트 트리 구간 쿼리 - 시간복잡도

이런 구간 쿼리의 사간 복잡도는 O(lg N)입니다.

직관적으로는 트리의 높이는 O(lg N)이니까 그럴거라고 생각할 수 있습니다.

혹은 쿼리 구간을 양 끝에서부터 세그트리의 구간들로 분해한다 치면 O(lg N)의 구간들로 쪼개질거니까 그럴거라고 생각할 수도 있습니다.

세그먼트 트리 구간 쿼리 - 시간복잡도

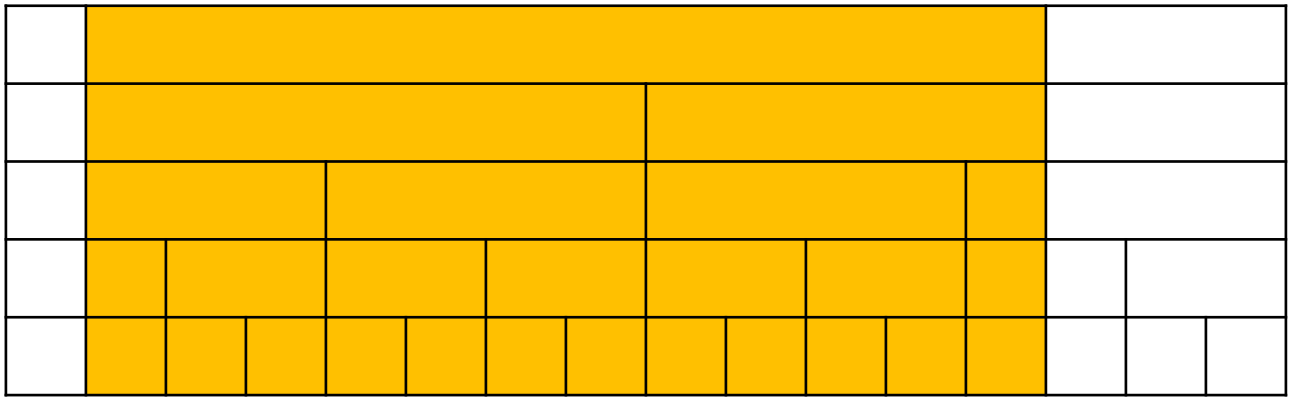
조금 더 그럴듯하게 증명해봅시다.

우리가 만든 트리의 높이는 O(lg N)입니다. 이 트리의 레벨별로 query 함수가 호출되는 노드가 최대 4개임을 증명하려고 합니다.

쿼리의 구간을 [l, r]이라고 합시다.

세그먼트 트리 구간 쿼리 - 시간복잡도

쿼리 구간이 연속적이므로 어느 레벨에서도 쿼리 구간에 완전히 포함되지 않는 노드는 최대 2개입니다.



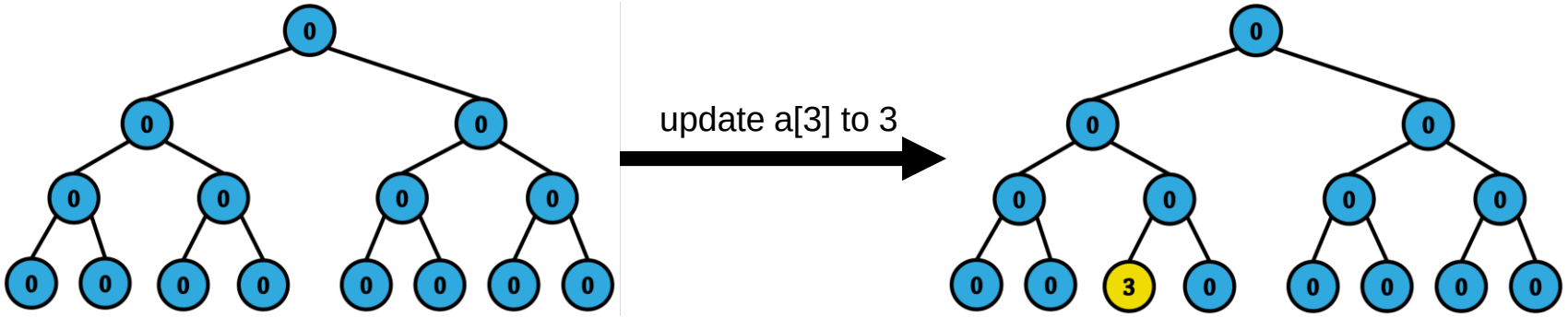
우리는 이런 노드들에 대해서만 재귀함수를 호출하기 때문에 각 레벨 별로 자식들에 대해서 함수를 호출 하는 횟수는 최대 4회입니다.

이것으로 구간 쿼리가 O(lg N)임을 알 수 있습니다.

세그먼트 트리 업데이트 쿼리

구간 합 쿼리까지 알아봤습니다. 특정 원소를 업데이트하는 쿼리도 알아봅시다.

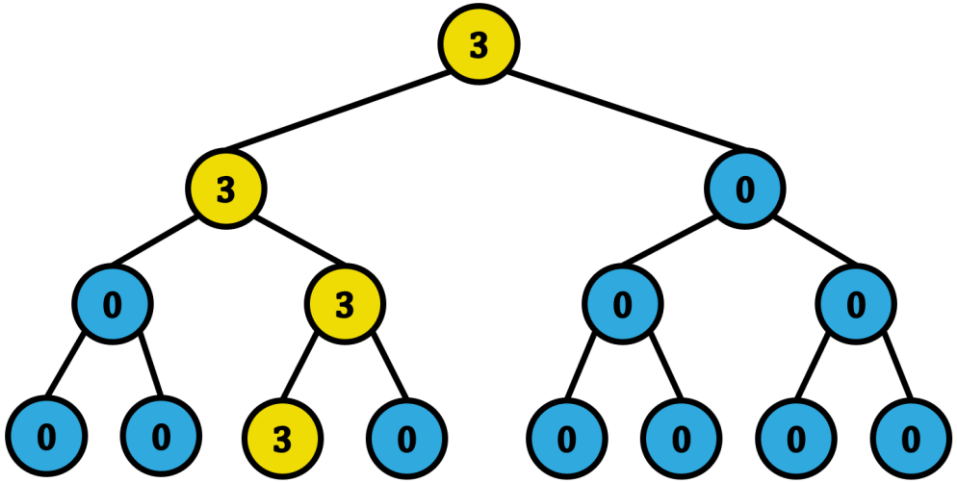
업데이트 쿼리는 간단합니다. 특정 원소를 업데이트한다는 것은 해당 원소에 맞는 리프노드를 변경하는 것과 동일합니다.



세그먼트 트리 업데이트 쿼리

리프 노드가 업데이트 되면 해당 리프 노드의 모든 조상들이 업데이트 되어야 합니다.

조상들의 개수는 O(lgN)이고 리프노드를 찾는 과정도 O(lgN) 으로 업데이트 또한 O(lgN)만에 가능합니다.



세그먼트 트리 업데이트 쿼리 - 구현

업데이트 또한 재귀적으로 구현할 수 있습니다.

```
void update(int node_idx, int node_left, int node_right, int update_idx, ll val) {
    if (node_left > update_idx || node_right < update_idx) return ;
    if (node_left == update_idx && update_idx == node_right) {
        tree[node_idx] = val;
        return ;
    }
    int mid = (node_left + node_right) / 2;
    update(node_idx * 2, node_left, mid, update_idx, val);
    update(node_idx * 2 + 1, mid + 1, node_right, update_idx, val);
    tree[node] = tree[node_idx * 2] + tree[node_idx * 2 + 1];
}
```

펜윅트리와 비재귀 형식의 세그먼트 트리

제가 소개한 구현 방식은 전부 재귀함수를 사용한 방식입니다.

세그먼트 트리를 비재귀 방식으로 구현할 수도 있으며 속도 면에서 재귀 방식보다 큰 이득이 있습니다.

또한 구간합을 구하는데 특화된 펜윅트리라는 자료구조도 있습니다. 이것도 상당히 빠릅니다.

여기선 안 다루지만 관심이 있는 분은 찾아보시면 좋을 거 같습니다.

백준 15816번 퀘스트 중인 모험가

퀘스트 중인 모험가 성공 출처

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
3 초	256 MB	602	169	81	20.823%

문제

모험가 x|최강영훈|x 은 온라인 RPG 게임 월드 아트 위그드라실(WAY)에서 활동중인 랭커이다.

x|최강영훈|x 은 매일 WAY의 퀘스트를 달성하는것에 재미를 느끼고 있다.

WAY에는 -10억부터 10억까지의 모든 정수 번호에 대해 퀘스트가 하나씩 부여되어 있다. 또한 특정 범위의 퀘스트를 모두 달성하면 업적을 받을 수 있다.

x|최강영훈|x 은 퀘스트를 달성하는 김에 업적도 얻고 싶었다. 하지만 WAY에는 항상 퀘스트를 순서대로 달성할 수 없는 경우가 존재해서 업적마다 몇 개의 퀘스트를 더 달성해야하는지 매번 계산해야 하는 불편함이 존재했다.

이를 안타깝게 여긴 모험가 사투-는 특정 범위의 퀘스트 중 모험가가 달성하지 못한 퀘스트의 개수를 출력하는 애드온을 개발하기로 했다.

물론 언제나 그렇듯, 구현은 당신이 해야 한다.

입력

첫째 줄에 지금까지 달성한 퀘스트의 개수 N이 주어진다. ($1 \le N \le 1,000,000$)

둘째 줄에 지금까지 달성한 퀘스트들의 번호 $Q_1 \dots Q_N$ 까지의 N개의 수가 주어진다. ($-1,000,000,000 \le Q[i] \le 1,000,000,000$, $Q[i] < Q[i+1]$)

셋째 줄에 애드온 요청의 개수 M이 주어진다. ($1 \le M \le 1,000,000$)

넷째 줄부터 M개의 줄에 걸쳐서 애드온에 요청할 명령이 주어진다.

1. 1 X: 퀘스트 번호 X를 달성했다. 애드온에 이를 반영해야 한다. ($-1,000,000,000 \le X \le 1,000,000,000$)
2. 2 L R: 퀘스트 번호 L 이상 R 이하인 퀘스트 중, 모험가가 달성하지 못한 퀘스트의 개수를 출력한다. ($-1,000,000,000 \le L \le R \le 1,000,000,000$)

백준 15816번 퀘스트 중인 모험가

만약에 주어지는 수들이 충분히 작다면 단순한 구간합 문제입니다.

그러나 쿼리로 주어지는 수들의 범위가 배열에 담기엔 너무 큽니다.

이럴 때 자주 사용하는 게 좌표 압축입니다.

백준 15816번 퀘스트 중인 모험가

만약 우리가 사용될 수를 전부 안다면 그런 수들을 정렬하고 상대적인 순서를 알 수 있습니다.

이를 토대로 수들을 상대적인 순서를 나타내는 수로 바꾸는 기법이 좌표압축 입니다.

방법은 간단합니다. 모든 수들을 정렬하고 중복하는 수들은 없앱니다.

그리고 각 수들을 자신의 정렬한 순서로 바꿔줍니다.

총 시간복잡도는 O(NlogN)입니다.

백준 15816번 퀘스트 중인 모험가

백준 18870번에서 좌표 압축에 대한 문제를 풀 수 있습니다. 아래는 18870번 AC 코드입니다.

```cpp
#include<bits/stdc++.h>
using namespace std;
int a[1000005];
int N;

int main() {
    cin.tie(nullptr); ios::sync_with_stdio(false);
    cin >> N;
    vector<int> comp;
    for(int i=0;i<N;++i) {
        cin >> a[i];
        comp.push_back(a[i]);
    }
    sort(comp.begin(), comp.end());
    comp.erase(unique(comp.begin(), comp.end()), comp.end());
    for(int i=0;i<N;++i) {
        int x = lower_bound(comp.begin(), comp.end(), a[i]) - comp.begin();
        cout << x << " \n"[i==N-1];
    }
    return 0;
}
```

http://boj.kr/117f3709c9c24771a0610bbf1cef1fbe

백준 15816번 퀘스트 중인 모험가

이제 문제에서 주어지는 수들을 전부 읽어들인 뒤에 좌표 압축을 진행하면 수의 범위가 줄어듭니다.

등장하는 수의 최대 개수는 300만개(=MAX)입니다.

따라서 쿼리당 O(lg MAX)에 처리할 수 있고 시간 내로 문제를 해결 가능합니다.

이와 같이 등장하는 수들을 좌표 압축해서 수를 변환한 뒤에 인덱스처럼 사용하는 유형은 굉장히 자주 등장합니다.

백준 15816번 퀘스트 중인 모험가

이와 같이 등장하는 수들을 좌표 압축해서 수를 변환한 뒤에 인덱스처럼 사용하는 유형은 굉장히 자주 등장합니다.

또한 위와 같이 문제에서 주어진 입력을 전부 다 읽은 다음에 해결할 수 있는 상황을 오프라인이라고 합니다.

일반적으로는 어려운 문제가 오프라인 상황에서 쉬워지는 경우도 많습니다.

백준 14287번 회사 문화 3

회사 문화 3 성공

☆

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
2 초	512 MB	651	320	225	46.778%

문제

영선회사에는 매우 좋은 문화가 있는데, 바로 상사가 직속 부하를 칭찬하면 그 부하가 부하의 직속 부하를 연쇄적으로 칭찬하는 내리 칭찬이 있다. 즉, 상사가 한 직속 부하를 칭찬하면 그 부하의 모든 부하들이 칭찬을 받는다.

이러한 내리 칭찬은 회사에 가장 큰 장점이 되었고, 사장 영선이는 이 장점을 이용하기 위하여 단 하루만 반대로 하기로 했다. 즉, 부하가 상사를 칭찬하면, 그 위로 쭉 사장까지 모두 칭찬을 받는다.

칭찬에 대한 정보는 실시간으로 주어진다.

입력으로 아래와 같은 쿼리가 주어질 것이다.

- `1 i w` : i번째 직원이 직속 부하 중 한 명으로부터 w만큼 칭찬을 받는다. ($1 \le i \le n$, $1 \le w \le 1,000$) 부하가 없다면 입력으로 들어오지 않는다.
- `2 i` : i번째 직원이 칭찬을 받은 정도를 출력한다.

직속 상사와 직속 부하관계에 대해 주어지고, 쿼리가 주어졌을 때 2번 쿼리에 따라 출력하시오.

입력

첫째 줄에는 회사의 직원 수 n명, 쿼리의 개수 m이 주어진다. 직원은 1번부터 n번까지 번호가 매겨져 있다. ($2 \le n, m \le 100,000$)

둘째 줄에는 직원 n명의 직속 상사의 번호가 주어진다. 직속 상사의 번호는 자신의 번호보다 작으며, 최종적으로 1번이 사장이다. 1번의 경우, 상사가 없으므로 -1이 입력된다.

다음 m줄에는 쿼리가 한 줄에 하나씩 주어진다.

백준 14287번 회사 문화 3

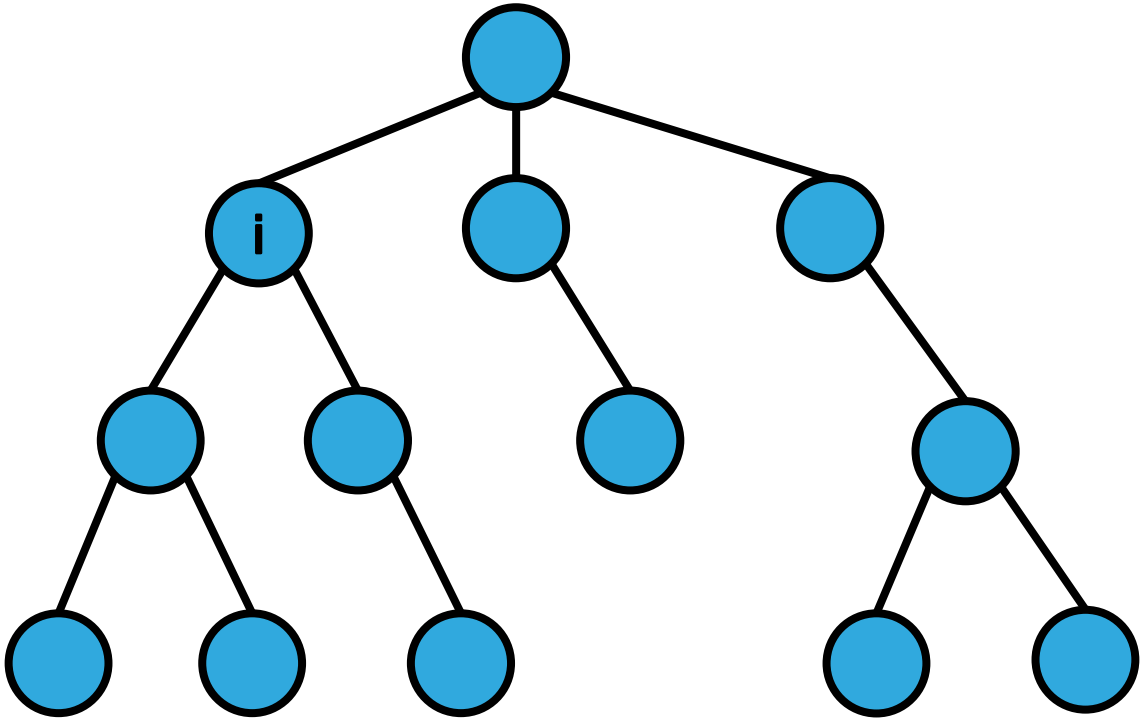
1번 쿼리가 들어오면 i번 노드의 모든 조상은 w만큼 칭찬을 받습니다.

이를 일일이 처리하면 O(N)입니다.

좀 바꿔서 생각해봅시다.

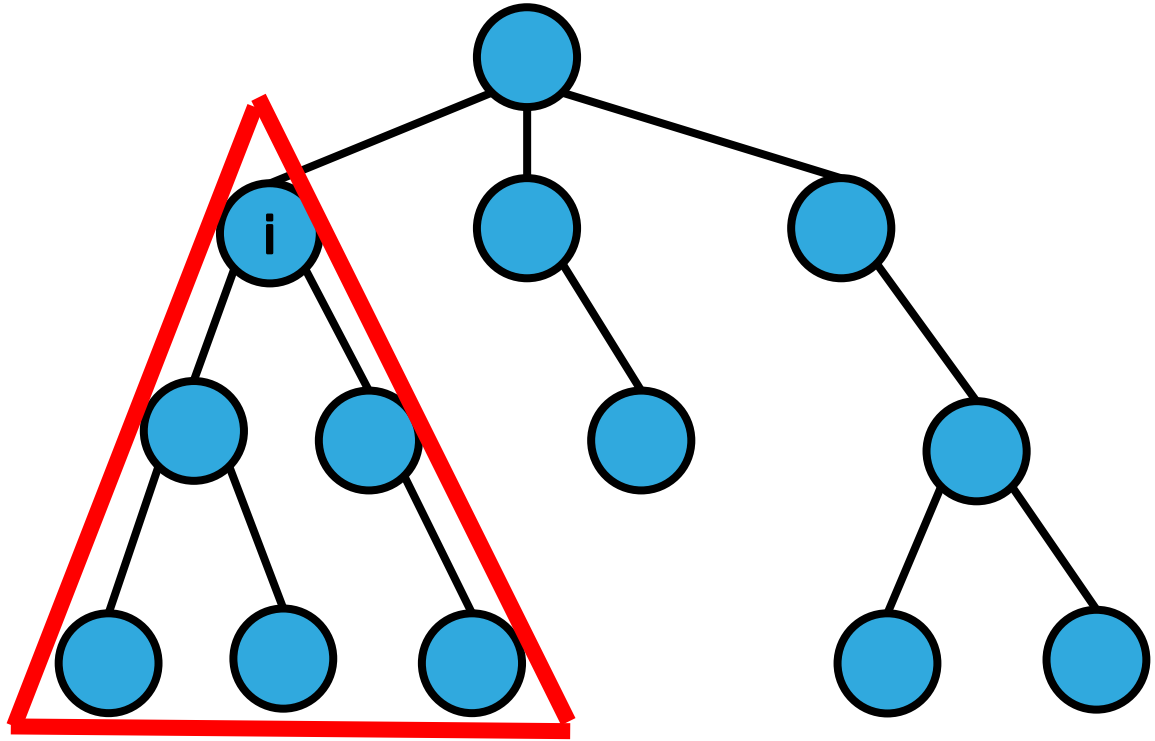
백준 14287번 회사 문화 3

i번 노드에게 칭찬을 할 수 있는 노드는 어떤 노드일지 생각해봅시다.



백준 14287번 회사 문화 3

i번 노드에게 칭찬이 가능한 노드는 i번 노드를 루트로 하는 노드들 말곤 없습니다.



백준 14287번 회사 문화 3

따라서, 2번 쿼리의 답은 i번 노드의 서브트리가 받은 칭찬의 합입니다.

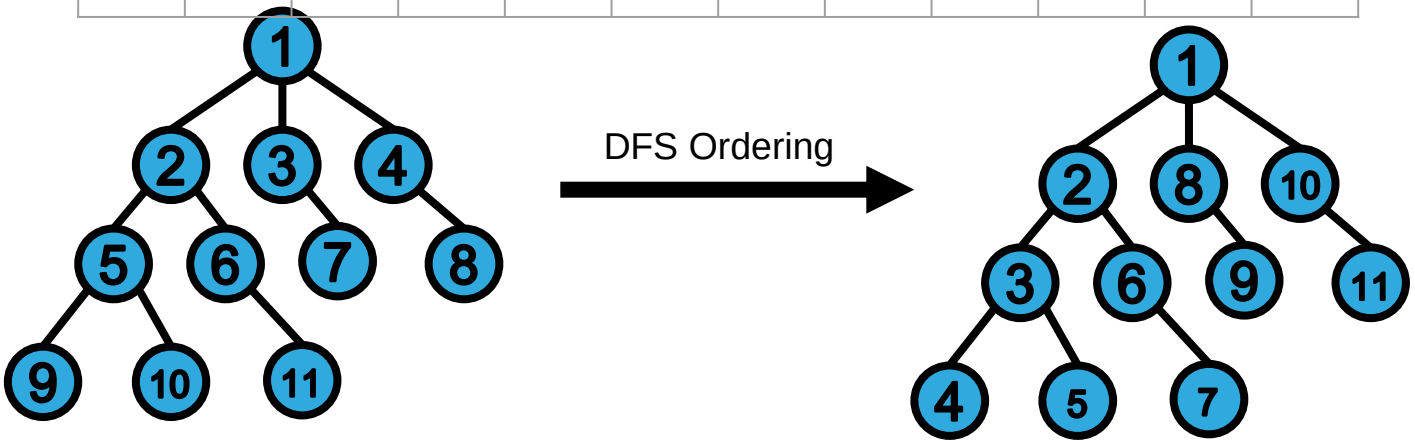
1번 쿼리는 O(1)으로 바뀌었지만 2번 쿼리가 O(N)입니다.

여기서 오일러 투어 테크닉이라는 것을 쓰면 빠르게 할 수 있습니다.

백준 14287번 회사 문화 3

오일러 투어 테크닉은 DFS Ordering을 통해서 트리를 배열처럼 만드는 테크닉입니다.
여기서 DFS Ordering이라는 것은 루트 노드에서부터 시작하는 DFS 순서대로 정점에 번호를 붙이는 것입니다.

기존	1	2	3	4	5	6	7	8	9	10	11
DFN	1	2	8	10	3	6	9	11	4	5	7

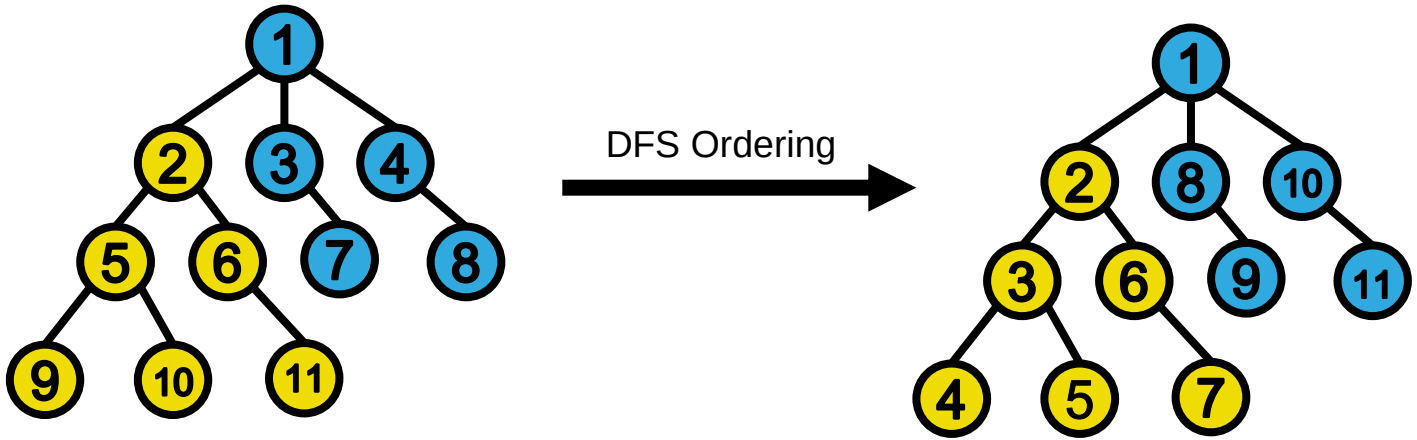


백준 14287번 회사 문화 3

DFS Ordering을 한 다음에는 어떤 노드의 서브트리는 DFS Ordering 상에서 연속하게 됩니다.
이를 통해서 트리를 배열과 같은 선형 모양으로 핀 효과를 가집니다.

기존	1	2	3	4	5	6	7	8	9	10	11
DFN	1	2	8	10	3	6	9	11	4	5	7

노드 2의
서브트리

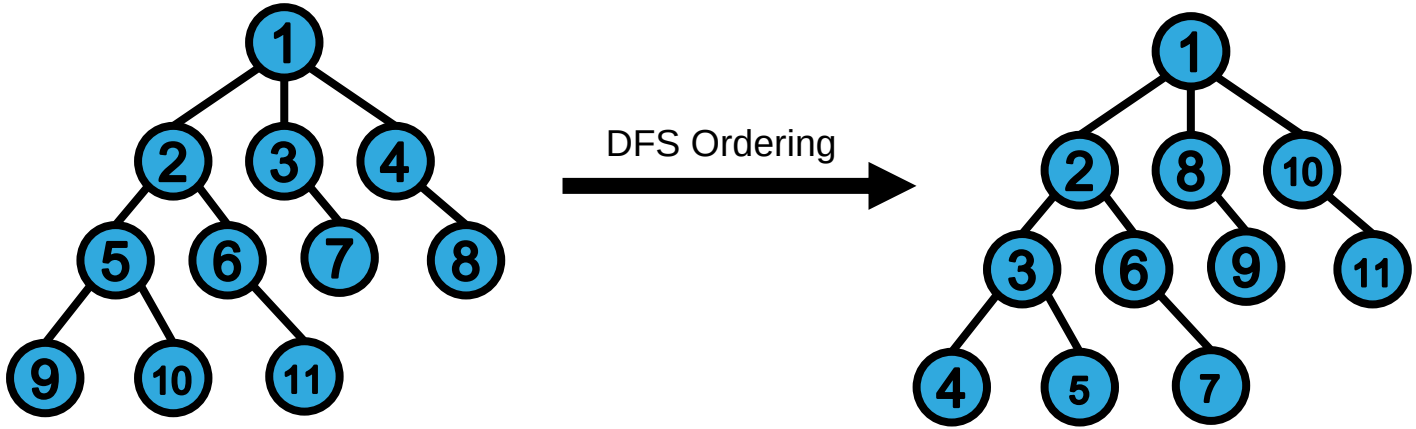


백준 14287번 회사 문화 3

아래 코드처럼 DFS 한번으로 DFS Ordering이 가능하고 [dfs_in[i], dfs_out[i]]이 i번 노드의 서브트리가 DFS order상 연속해 있는 구간이 됩니다.

```
int dfs_in[100005];
int dfs_out[100005];
int dfn;
vector<vector<int>> G(100005);

void dfs(int cur) {
    dfs_in[cur] = ++dfn;
    vis[cur] = true;
    for(int nxt:G[cur]) {
        if(!vis[nxt]) dfs(nxt);
    }
    dfs_out[cur] = dfn;
}
```



백준 14287번 회사 문화 3

세그먼트 트리는 배열에 대해서 사용이 가능했고 이제 트리의 서브트리에도 사용할 수 있게 됐습니다.

이를 통해서 O(lg N)에 2번 쿼리가 해결이 가능하고 1번 쿼리의 업데이트 또한 O(lg N)에 해결이 가능합니다.

백준 19646번 Random Generator

Random Generator 성공 출처

☆

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
1 초 (추가 시간 없음)	1024 MB	190	123	99	64.706%

문제

국렬이는 1부터 N까지의 양의 정수로 이루어진 순열을 주어진 양의 정수 $w_1 \dots , w_N$를 이용해서 무작위로 만들 것이다. 다음은 무작위로 순열을 만드는 방법이다.

1. 1부터 N까지의 양의 정수 i ($1 \le i \le N$)를 연속적으로 w_i개씩 배치한다.
2. 현재 배치된 양의 정수의 총 개수를 W라고 하자. 1부터 W까지의 양의 정수들 중에서 균등하게 숫자 하나 p_i를 선택한다.
3. p_i번째 수를 순열에 추가한다.
4. 순열에 추가한 수들을 전부 지우고, 남은 수가 없을 때까지 2부터 4의 과정을 거친다.

w_1부터 w_N까지, p_1부터 p_N까지의 수가 주어졌을 때, 나오는 순열을 구해 보자.

입력

첫 번째 줄에는 N ($1 \le N \le 200,000$)이 주어진다.

두 번째 줄에는 1,000 이하의 양의 정수 w_1부터 w_N까지가 주어진다.

세 번째 줄에는 양의 정수 p_1부터 p_N까지가 주어진다. p_1부터 p_N까지로 수열을 만들 수 있는 경우만 주어진다.

출력

최종적으로 나오는 1부터 N까지 양의 정수들로 이루어진 수열을 구하자.

백준 19646번 Random Generator

우리가 해결해야 되는 상황은 다음과 같습니다.

i가 w_i개씩 있을 때 전체에서 p_i번째의 수를 찾아야 합니다.
찾은 뒤에는 모든 i를 지웁니다.
이 과정을 더 이상 수가 없을 때까지 반복합니다.

1 2 2 3 3 3 4 4 4 4 5 5 5 5 5 -> 6번째 수(3)를 찾고 지운다.
1 2 2 4 4 4 4 5 5 5 5 5 -> 4번째 수(4)를 찾고 지운다.
1 2 2 5 5 5 5 5 -> 6번째 수(6)를 찾고 지운다.
1 2 2 -> 1번째 수(1)를 찾고 지운다.
2 2 -> 2번째 수(2)를 찾고 지운다.
종료

백준 19646번 Random Generator

우리가 해결해야 되는 문제는 수열에서 p_i번째 수를 찾고 그 수를 전부 지우는 작업을 빠르게 하는 것입니다.

문제에서 제시할 수열은 항상 오름차순을 유지하기 때문에 p_i번째 수를 찾는 것은 p_i번째로 작은 수를 찾는 것과 동일합니다.

이는 세그먼트 트리를 이용하면 해결이 가능합니다.

백준 19646번 Random Generator

수열에 존재하는 수들이 1과 N 사이의 값을 가지기 때문에 이 값을 인덱스로 하고 이 값의 개수를 값으로 가지는 배열을 생각합시다.

이는 입력으로 주어지는 w_i와 동일한 배열입니다. w_i의 누적합을 f[i]라고 합시다.

p_i번째 수는 f[k-1]<p_i이면서 $p_i \leq$ f[k]인 k입니다.

백준 19646번 Random Generator

배열이 변하지 않는다면 누적합 배열을 한번 만들고 이분탐색해서 해결할 수 있습니다.

하지만 한 번 추가한 수는 전부 지운다는 조건이 있었으므로 세그먼트 트리를 사용해서 누적합 배열을 관리할 필요가 있습니다.

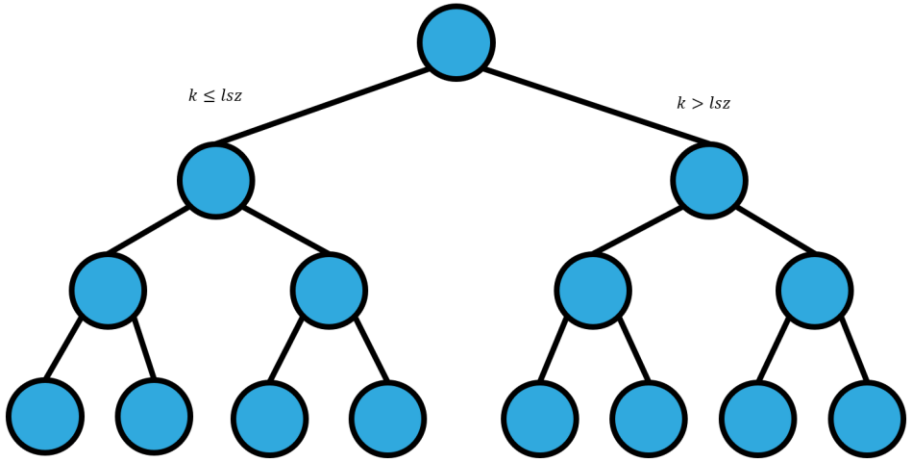
누적합을 한 번 구하는 데에 O(lg N), 이분 탐색에 O(lg N)이 걸립니다.

따라서, 총 시간복잡도는 $O(\lg^2 N)$이 됩니다.

백준 19646번 Random Generator

이것보다 빨리 할 수 있는 방법이 있습니다. 세그트리를 탐색할 때 현재 서브트리에서 k번째 원소가 왼쪽에 있을지 오른쪽에 있을지를 알 수 있습니다.

왼쪽 자식 노드의 값을 lsz이라고 합시다. k가 lsz 보다 작거나 같을 땐 k번째 원소는 왼쪽에 위치해 있습니다. 반대로 더 클 땐 오른쪽에 위치해 있습니다.



백준 19646번 Random Generator

따라서 왼쪽에 있을 땐 왼쪽으로 내려가고 오른쪽에 있을 땐 오른쪽으로 내려가며 탐색을 진행하면 됩니다.

오른쪽으로 진행할 땐 k에서 lsz을 빼줄 필요가 있습니다.

```
int kth(int node, int s, int e, int k) {
    if(s == e) return s;
    int lsz = tree[node*2];
    int mid = (s+e) >> 1;
    if(k <= lsz) return kth(node*2, s, mid, k);
    else return kth(node*2+1, mid+1, e, k-lsz);
}
```

Problem Set

- 필수 문제
  - 15816 퀘스트 중인 모험가
  - 14287 회사 문화 3
  - 19646 Random Generator
- 연습 문제
  - 12837 가계부 (Hard)
  - 11505 구간 곱 구하기
  - 14428 수열과 쿼리 16
  - 18436 수열과 쿼리 37
  - 1306 달려라 홍준
  - 1777 순열복원
  - 2243 사탕상자
  - 17400 깃발춤
  - 1280 나무 심기

Problem Set

- 연습 문제
  - 2517 달리기
  - 3653 영화 수집
  - 5419 북서풍
  - 13537 수열과 쿼리 1
  - 11658 구간 합 구하기 3
  - 17627 XORanges
  - 15899 트리와 색깔
  - 17408 수열과 쿼리 24
  - 18227 성대나라의 물탱크
  - 2336 굉장한 학생
  - 11672 Guessing Camels
  - 7737 삼각분할
  - 16532 Looking for the Risk Factor
  - 16993 연속합과 쿼리
  - 17076 망가진 데이터

“Any Question?”