#11-1. 이진 탐색 트리

나정휘
https://justiceHui.github.io/

이진 탐색 트리

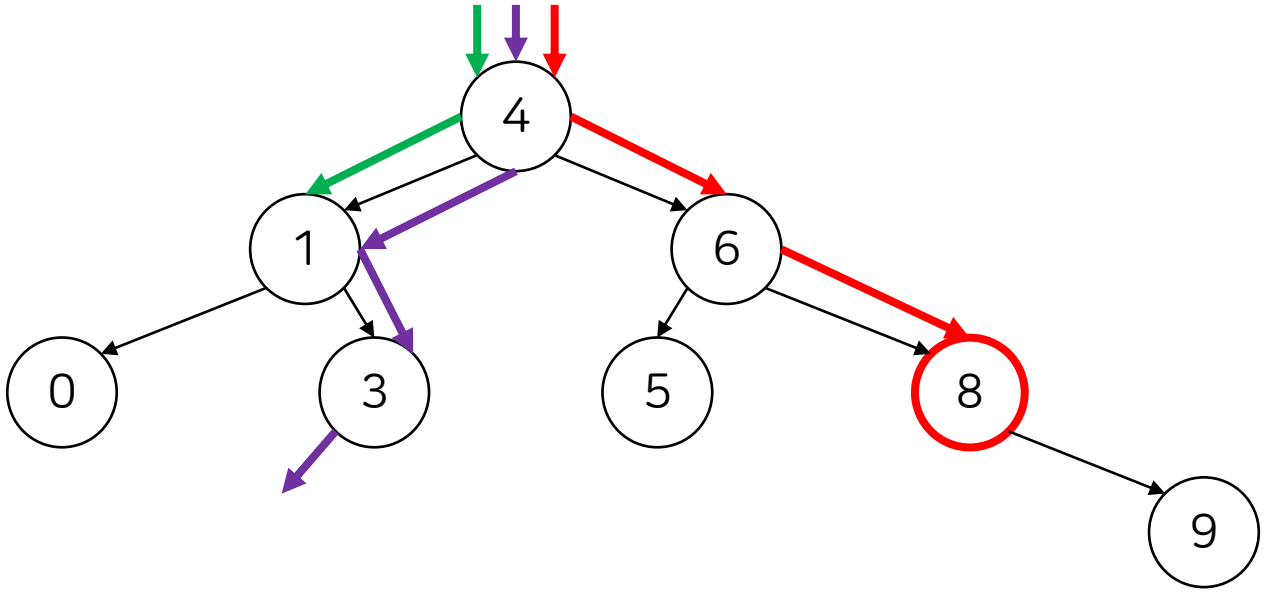
이진 탐색

- 정렬된 구간 [l, r]에서 어떤 값 x가 있는지 확인하는 알고리즘
  - 중간 지점 m = (l+r)/2를 잡고
  - A[m] = x이면 탐색 완료
  - A[m] < x이면 x는 m보다 오른쪽에 있음 : l = m + 1
  - A[m] > x이면 x는 m보다 왼쪽에 있음 : r = m − 1
- A = {0, 1, 3, 4, 5, 6, 8, 9}, x = 8
  - l = 0, r = 7, m = 3, A[m] = 4
  - l = 4, r = 7, m = 5, A[m] = 6
  - l = 6, r = 7, m = 6, A[m] = 8, 종료
- x = 1
  - l = 0, r = 7, m = 3, A[m] = 4
  - l = 0, r = 2, m = 1, A[m] = 1, 종료

이진 탐색 트리

이진 탐색 트리 (Binary Search Tree)

- 이진 탐색 과정을 이진 트리로 나타낸 것
  - x = 8
  - x = 1
  - x = 2
- 할 수 있는 연산
  - 트리에 원소 삽입/삭제
  - 어떤 원소가 있는지 확인
  - 가장 큰/작은 원소
  - x 이상/초과인 가장 작은 원소
  - x 이하/미만인 가장 큰 원소
  - 모든 원소를 오름차순으로 순회

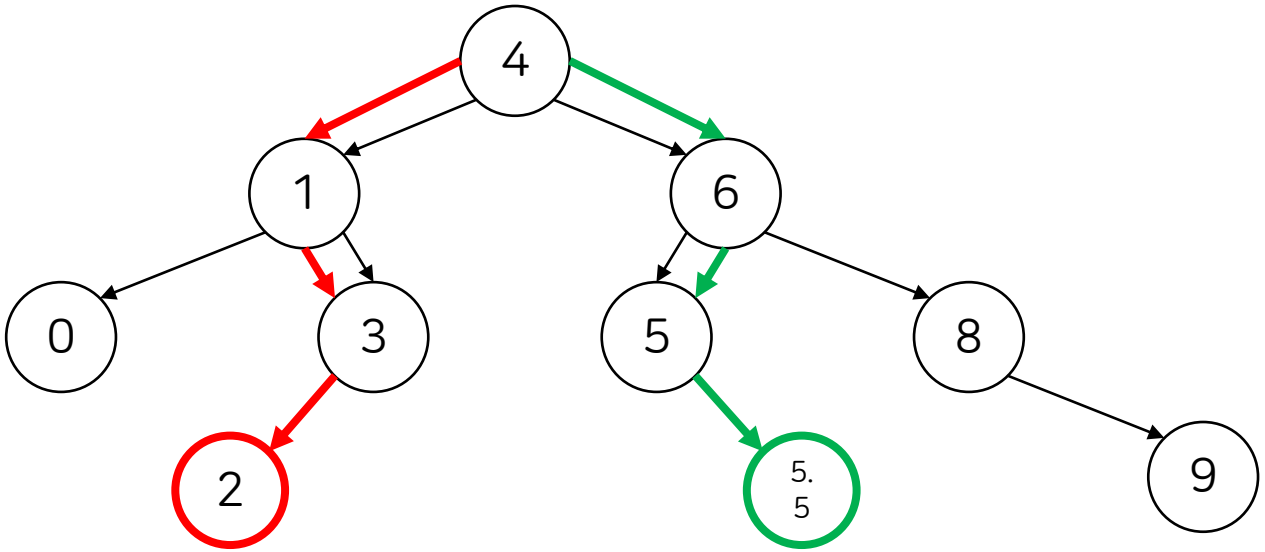


이진 탐색 트리

원소 삽입

- 루트부터 시작
- 현재 정점의 원소보다 크면 오른쪽 / 작으면 왼쪽으로 이동
- 더 이상 내려갈 수 없을 때까지 이동
- 마지막으로 방문한 정점의 자식으로 추가

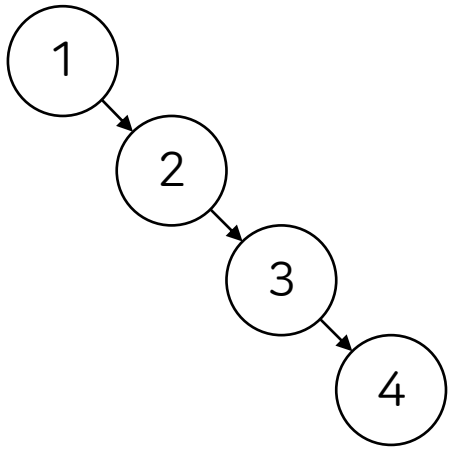
- INSERT 2
- INSERT 5.5



이진 탐색 트리

원소 삽입

- 시간 복잡도: 트리의 높이를 h라고 하면 O(h)
- 최악의 경우 O(N)
  - INSERT 1
  - INSERT 2
  - INSERT 3
  - ...
- 최선의 경우 O(log N)



이진 탐색 트리

원소 삭제

- 삭제해야 할 정점 x를 찾음
- 리프 정점이면 그냥 삭제
- 자식 정점이 1개면 x의 부모와 자식을 연결하고 x 삭제
- 자식 정점이 2개면
  - 오른쪽 서브 트리에서 가장 작은 원소 m을 찾음 (x보다 크면서 가장 작은 원소)
  - m을 x 자리로 옮기고 m을 삭제 (m은 자식 정점이 1개 이하)

- 시간 복잡도: O(h)

질문?

이진 탐색 트리

가장 큰/작은 원소

- 가장 큰 원소: 루트에서 시작해서 오른쪽 자식으로 계속 내려가면 됨
- 가장 작은 원소: 루트에서 시작해서 왼쪽 자식으로 계속 내려가면 됨

- x보다 큰 원소 중 가장 작은 원소
  - x의 오른쪽 자식이 존재하면 오른쪽 서브 트리에서 가장 작은 원소
  - x의 오른쪽 자식이 없으면 x의 조상을 따라가면서 처음으로 만나는 x보다 큰 원소
    - 현재 정점이 왼쪽 자식일 때까지 부모 정점을 따라가면 됨

- x보다 작은 원소 중 가장 큰 원소
  - 반대로 하면 됨

이진 탐색 트리

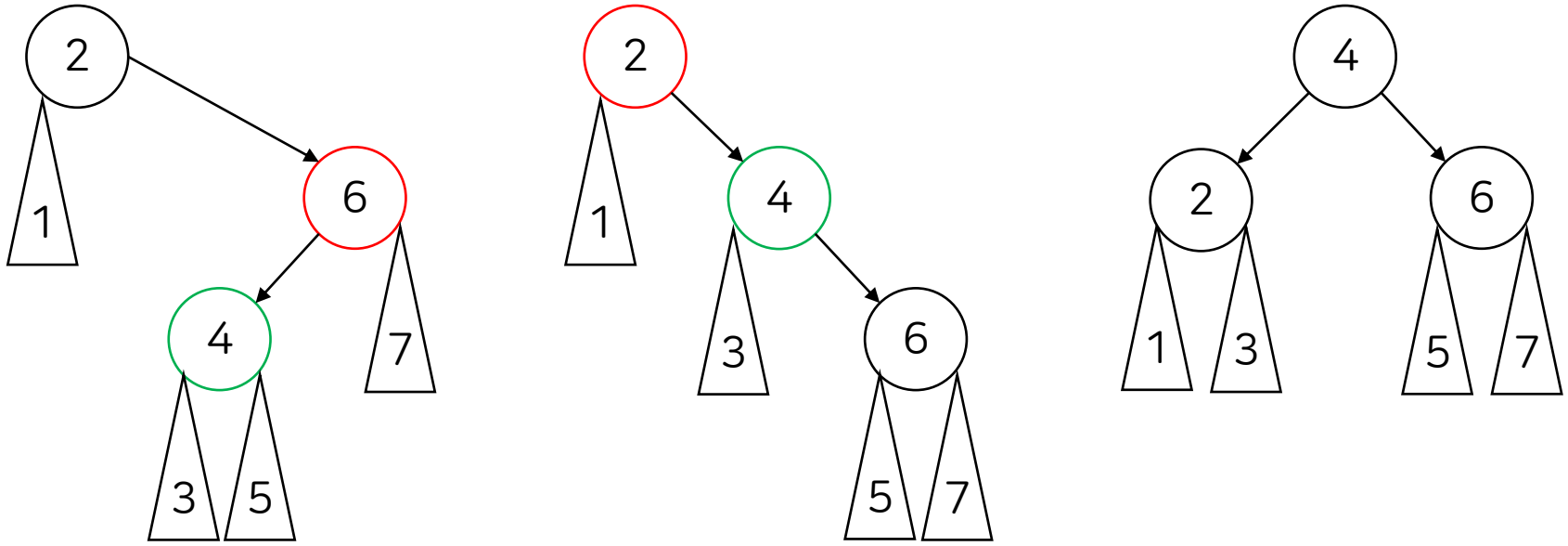
중위 순회 (Inorder Traversal)

- 이진 탐색 트리의 중위 순회는 원소를 오름차순으로 순회함
  - 왼쪽 서브 트리(자신보다 작은 원소)를 모두 순회한 뒤
  - 루트(자신)을 보고
  - 오른쪽 서브 트리(자신보다 큰 원소)를 모두 순회함

이진 탐색 트리

균형 잡힌 이진 탐색 트리 (Balanced Binary Search Tree)

- 단순하게 구현하면 최악의 경우 O(N)을 피할 수 없음
- BBST : 높이를 O(log N)으로 유지하는 BST
  - 대충 이런 느낌으로 BST 성질을 만족시키면서 트리를 바꿈
  - AVL Tree, Red Black Tree, Treap, Splay Tree, ...



질문?

이진 탐색 트리

std::set

- #include <set>
- 집합을 관리하는 컨테이너
  - 원소의 중복을 허용하지 않음 (std::multiset은 중복 허용)
  - 원소를 오름차순으로 관리
  - 원소 삽입/삭제/검색
  - 가장 큰/작은 원소
  - x 이상/초과인 가장 작은 원소
  - 오름차순/내림차순 순회
- 보통 Red Black Tree로 구현되어 있음
  - 모든 연산은 O(log N)에 동작
  - 오름차순/내림차순 순회는 O(N)

이진 탐색 트리

std::map

- #include <map>
- key-value 쌍을 관리하는 컨테이너
  - key의 중복을 허용하지 않음
  - value 변경 가능
  - key에 대해서 std::set의 모든 연산 수행 가능

이진 탐색 트리 – 예시 1

BOJ 1822 차집합

- 집합 A, B가 주어지면 A – B의 원소를 오름차순으로 출력하는 문제

- std::set 사용하면 됨
  - A.find(x) != A.end() 대신 A.count(x) 사용해도 됨
  - count는 multiset에서 시간 복잡도가 원소 개수에 비례하므로 조심

```cpp
#include <bits/stdc++.h>
using namespace std;

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    int N, M; cin >> N >> M;
    set<int> A;
    for(int i=0; i<N; i++){
        int x; cin >> x;
        A.insert(x);
    }
    for(int i=0; i<M; i++){
        int x; cin >> x;
        if(A.find(x) != A.end()) A.erase(x);
    }
    cout << A.size() << "\n";
    for(auto i : A) cout << i << " ";
}
```

이진 탐색 트리 – 예시 2

BOJ 20920 영단어 암기는 괴로워

- N개의 단어가 주어지면 특정 기준으로 정렬해야 함
  - 많이 등장한 문자열을 앞에 배치
  - 길이가 긴 단어를 앞에 배치
  - 사전 순으로 앞에 있는 단어를 앞에 배치

- std::map 사용해서 단어의 등장 횟수를 셀 수 있음
  - map에 존재하지 않는 key에 대한 value는 0으로 초기화

```cpp
#include <bits/stdc++.h>
using namespace std;

int N, M;
vector<string> V;
map<string, int> C;

bool Compare(const string &a, const string &b){
    if(C[a] != C[b]) return C[a] > C[b];
    if(a.size() != b.size()) return a.size() > b.size();
    return a < b;
}

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    cin >> N >> M;
    for(int i=0; i<N; i++){
        string s; cin >> s;
        if(s.size() >= M) V.push_back(s), C[s]++;
    }
    sort(V.begin(), V.end(), Compare);
    for(int i=0; i<V.size(); i++){
        if(i == 0 || V[i-1] != V[i]) cout << V[i] << "\n";
    }
}
```

이진 탐색 트리 – 예시 3

BOJ 2015 수들의 합 4

- 양의 정수로 구성된 배열 A가 주어짐
- A[i] + A[i+1] + … + A[j-1] + A[j] = K가 되는 경우의 수를 구하는 문제

- 누적 합 배열을 만들면 S[j] – S[i-1] = K인 (i, j)의 개수를 구하는 문제
- 각각의 j에 대해, S[i-1] = S[j] – K 인 i의 개수를 세면 됨
- std::map을 사용해 S[j] – K의 등장 횟수를 관리할 수 있음

```cpp
#include <bits/stdc++.h>
using namespace std;
using ll = long long;

ll N, K, A[202020], S[202020], R;
map<int, int> C;

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    cin >> N >> K;
    for(int i=1; i<=N; i++) cin >> A[i];
    for(int i=1; i<=N; i++) S[i] = S[i-1] + A[i];
    for(int i=0; i<=N; i++) R += C[S[i]-K], C[S[i]] += 1;
    cout << R;
}
```

질문?