알고리즘 검색 정리

강주한

**1,정렬과 검색의 차이**

정렬 : 정적 데이터, 삽입/삭제 불가능, 배열 사용, 연산 값 비쌈(검색에 비해)

검색 : 동적데이터, 삽입/삭제 가능, (트리, 해싱) 사용, 연산 값 쌈(정렬에 비해)

**2.검색을 하는 방법과 구성요소**

검색을 할 때는 “키”를 사용한다. 키를 사용하여 밸류를 찾는다.

이러한 방식을 “심볼 테이블”이라고 한다.

**키 –밸류 쌍 추상화** -> 각 **레코드**는 **키와 밸류의 쌍**

**맵(map)/딕셔너리(Dictionary)/연관배열(associative array)**

키 : 검색에 사용하는 필드(중복 불가)

밸류: 관련(부속,위성) 데이터

EX)학번 -> 이름, 카드번호 -> 이름,휴대폰,주소 등등

201737003 -> 강주한

**2.검색의 3요소**

검색 : 주어진 키에 대하여 검색한 후, 해당 밸류를 반환

삽입 : 키와 함께 밸류를 삽입 – 만약, 키가 이미 존재하면, 밸류를 업데이트

삭제 : 특정 키를 검색하여 해당 키- 밸류 쌍을 삭제, 키만 검색하면 삭제 가능

**3. 배열과 연관배열의 차이**

배열은 인덱스(index)를 통해 배열 내 모든 밸류들에 접근한다.

연관배열은 키(key)를 통해 값들에 접근한다.

EX) a[3] => 배열, a[“201737003”] => 연관배열

**4. map vs set**

Map은 키로 값들을 검색하는 것. Set은 특수한 경우에 사용

Map의 경우 배열의 인덱스대신 Key를 사용해 검색한다고 생각하면 된다.

**5. 이분검색이란?**

**선형자료구조**를 이용한 심볼 테이블 구현과 검색

선형자료구조란? **연결리스트, 배열** 등 데이터를 한 줄로 세우는 것.

연결리스트와 배열의 차이점은 배열은 메모리내에서도 데이터들이 바로 옆에 붙어있는 반면 연결리스트의 경우 서로 떨어져 있지만 다음 데이터를 가리키는 방식을 이용하는 것.

검색 : 순차검색(앞에서부터 하나하나 확인)

삽입 : 키를 검색하여 성공하면 해당 밸류를 업데이트, 실패하면 맨 앞에 삽입

검색/삽입,삭제의 최악의 경우 시간복잡도 : O(n)이다.

평균의 경우에도 : O(n)이다. 하지만, O(n/2)정도이다.

**정리를 안 한 선형자료구조**를 이용한 심볼 테이블의 검색의 경우, 검색,삽입,삭제 **모두 O(n)의 선형시간이 걸린다.**

**2) 이분검색이란,**

이분 검색이란 Binary Search를 뜻하는 것. 정렬되어 있는 경우 반 씩 나누어 검색 하고자 하는 값의 경우보다 작은 경우는 왼쪽으로 큰 경우는 오른쪽으로 나누며 검색하면 되기 때문에 시간복잡도가 T(n) = T(n/2) + c이다. 이를 마스터 정리를 사용하면 a = 1, b = 2, k = 0이므로 O(log n)이 나오게 된다.

이분검색의 최악의 경우 시간복잡도가 O(log n)이다.

**이분 검색의 수도 코드 :**

binarySearch(a,x,low,high)

if(low > high) return -1 //검색하고자 하는 값이 없으면 -1 반환

mid = (low + high) / 2

if(x==a[mid].key) return mid;

if(x < a.[mid].key) return binarySerch(a,x,low,mid – 1) //왼쪽 검색

return binarySearch(a,x,mid + 1, high)

무조건, 정렬하여 검색하는 것이 좋은 것은 아니다. 만약, 선형적 자료구조는 검색/삽입/삭제에 선형시간이 걸린다.

트리 -> BST(Binary Search Tree) 이진 검색 트리(평균적으로 O(log n), 최악 O(n)

대신, 균형 잡힌 트리 -> 높이가 작은 트리일수록 검색의 속도가 빠르다.

RedBlackTree(무슨 일이 있어도 균형 잡힌 트리) -> O(log n)

해싱 -> 검색속도가 무려 O(1)이다.(amortized analysis) 대신 검색만 가능(삭제 삽입 불가능)

**트리를 사용함**으로써 **선형구조 => 계층구조로** 표현 가능해진다.

(부모 자식 관계)

**6. 트리의 구성요소**

**트리** : 모든 노드들이 연결된 계층구조(부모,자식)

**잎 노드** : 자식노드가 모두 null인 노드를 뜻함.

**이진트리** : 모든 노드가 2개의 섭 트리(공집합 가능)을 가지는 트리

트리의 경우, 왼쪽 자식들은 모두 자기 자신의 ‘키’보다 작고 오른쪽의

자식들‘키’는 자기 자신 ‘키’보다 크다. 이런 조건을 만족하는 경우가 이진 검색

트리이다.

트리는 보통 재귀를 통해 구현한다.