-정렬알고리즘(몇번수행만에 정렬완료, 정렬이미완료 재정렬반복

BUBBLE + SELECTION

insertion sort	다른 데이터를 정렬된 데이터들 내부로 삽입. 이미 정렬된 데이터 인접된 자리로 이동
shell sort (GAP = 4)	다른 데이터를 정렬된 데이터들 내부로 삽입. 이미 정렬된 데이터 멀리 떨어진 자리로 이동
quick sort	정렬되지 않은 데이터들을 일정 기준으로 분할 분할 상태에서 정렬 PIVOT-1번/마지막/중간.. 작은거값,,,PIVOT값,,,,큰값
merge sort	정렬되지 않은 데이터들을 일정 기준으로 분할 합병 상태에서 정렬 N/2 ... 1개 분할 합병 2 / 4 / 8 .../N
counting sort	다른 데이터 배열의 값을 인덱스로 하여 정렬 특정값 빈도수 도수분포표 1. 데이터범위 한정 2. 별도 배열 1개 필요

- 트리

1>부모 자식 형제등의 계층 구조로 데이터를 표현

2>표현 데이터 각각은 node

3>root node로부터 시작

4>leaf node로 끝남

5> 부모는 여러개 자식 가짐

6> 자식은 1개의 부모 가짐

- 이진트리

1>위 트리 구조 가운데 자식을 1-2개로만 제한하여 가지는 트리 구조.

2>왼쪽 자식과 오른쪽 자식을 구분.

3> 트리 조회시

3-1. level 조회-BREADTH FIRST SEARCH(BFS)

3-2. depth first 조회(DFS)

3-2-1 inorder – 왼쪽자식 – 부모 – 오른쪽자식

3-2-2 preorder –부모 – 왼쪽자식 – 오른쪽자식

3-2-3 postorder –왼쪽자식 –오른쪽자식 -부모

- 이진검색트리(binary search tree – BST)

1>위 이진 트리 구조 가운데 조건 부여한 구조.

2>부모보다 작은 값을 가지는 노드는 왼쪽 자식으로 배치

3>부모와 부모보다 큰 값을 가지는 노드는 오른쪽 자식으로 배치

4> 같은 키를 가지는 노드 존재X.

5> 이진검색트리

6>트리 조회시 inorder – 왼쪽자식 – 부모 – 오른쪽자식 조회시 오름차순 정렬

7> 이진검색트리 구성 구현

8> 검색(특정데이터)/데이터추가/데이터삭제 구현

Node 생성 -

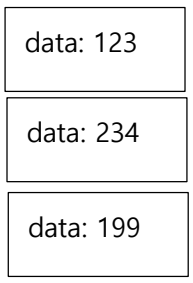
```
class Node{
int key;-> 각 노드 구분식별자(중복X)
Node left;
Node right;
}
class Node{
String key;
Node left;
Node right;
}
class Node<T>{
T key;
Node left;
Node right;
}
```

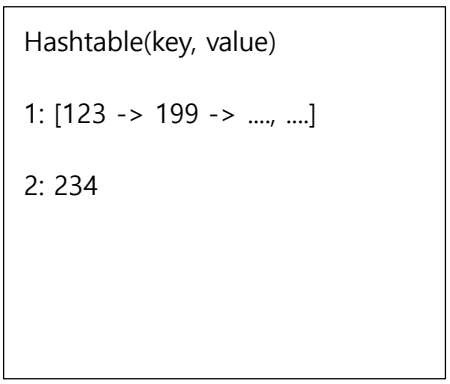
-검색종류

1.연결리스트검색	**삽입/삭제 빠르다**
2.이진검색트리	**반으로 줄어듬** **검색속도 빠르다** **삽입/삭제 그냥 그렇다-트리 재구성**
3.배열검색	**3-1. 선형검색 – 3장** **3-2. 이진검색-정렬된 상태 조건 – 3장** **검색속도 빠르다** **3-3. 해시검색** **검색속도 / 삽입삭제 효율적**

- 해시검색

해쉬법





1> 해시값을 생성하면 다른 데이터에 대하여 해시값 충돌

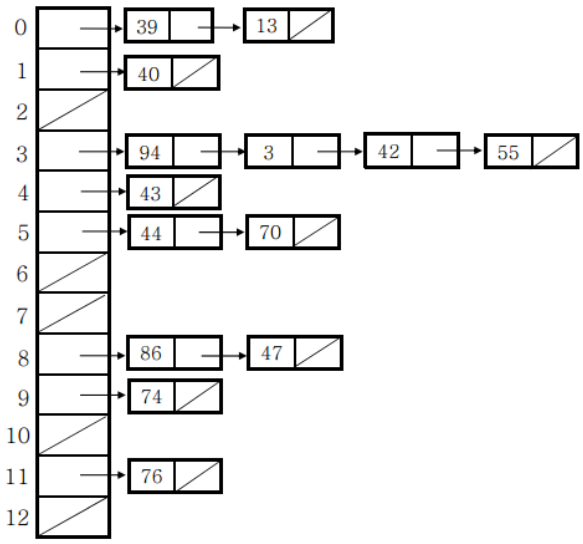
2> 충돌 피하자

3> 같은 해시값 – 배열 /list..

체인법 chaining

체인법이란 해시값이 같은 데이터를 체인 모양의 연결 리스트로 연결하는 방법을 말하며 오픈 해시법이라고도 한다.

배열의 각 버킷(해시 테이블)에 저장하는 것은 인덱스를 해시값으로 하는 연결 리스트의 앞쪽 노드(**head node**)를 참조하는 것이다.



체이닝 예시

class Node{

int key;

int value;

Node next;

}

Main

insert / remove / find /display