* Data File을 찾는 것 보다 인덱스 File을 찾는 것이 더 빠르다면 인덱스를 쓰는 것이 더 빠르다.

문제점 : (심지어 인덱스를 사용하지 못할만큼)

회원 관리를 위한 파일 => 오픈하는날(회원 : 0명) => 새로운 값(회원)을 넣는다. => 인덱스를 만들지 않음..? => 인덱스가 없음? => ????

Search Algorithm => Insert Algorithm을 했을 때 생각하지 못했던 문제 발생 => Search Algorithm 수정 => Delete Algorithm을 했을 때 생각하지 못했던 문제 발생 => Insert Algorithm 수정 => Search Algorithm 수정 => 계속 유기적으로 바뀜.

여러 번 반복되는 알고리즘은 문제가 있다.

속도 비교 == Disk Access 횟수 비교

검색시 O(n) => n은 디스크 블록의 개수

* 데이터의 양이 많더라도 그것이 꼭 블록의 개수가 되는 것은 아니다. 어떻게 블록에 적재하는가에 따라서 블록의 개수의 차이가 있다.

데이터(레코드)의 양의 영향을 미치는 요소 => 과연 무엇인가?

* 인덱스의 개수 <= 디스크 블록의 개수

Delete를 할 때마다 인덱스를 바로바로 지울 것 => 최악의 경우에도 같아지고 디스크 블록의 개수가 더 많다.

결국 인덱스의 구조를 이용하는 것이 더 빠르다…

하지만, 신경써야할 것이 너무 많기 때문에 쓰지는 않는다.

* 적합한 인덱스 구조

조건

1. 키 필드의 중복이 있느냐? 없느냐?
2. 정렬이 되어 있느냐? 안되어 있느냐?

* 중복 O 정렬 O
* 중복 O 정렬 X
* 중복 X 정렬 O
* 중복 X 정렬 X

과제 : 인덱스를 만들어 보자 ppt에 있는 그림 (일반적인 인덱스 구조) 처럼 구조를 만들어보고 그 구조를 설명하자. (다음주 화요일 까지)