6주차 3차시 알고리즘의 소개

【학습목표】

1. 정렬 알고리즘의 개념과 종류에 대해서 살펴보고 설명할 수 있다.
2. 탐색 알고리즘의 개념과 문제 해결방식을 살펴보고 설명할 수 있다.

학습내용1 : 정렬 알고리즘

- 정보를 처리하기 위해서는 어떤 기준에 따라 순서가 정해져야 처리 속도가 빨라진 다는 점을 강조

1. 정렬(Sorting) 알고리즘

* 정렬 알고리즘이란?
- 모든 자료를 처리할 때 꼭 사용하는 방식
- 정렬방식은 키(Key)를 중심으로 오름차순 또는 내림차순으로 정렬
- 정렬 대상 데이터의 유형은 숫자, 문자, 이미지와 같은 멀티미디어 자료 등
- 정렬에는 내부정렬(알고리즘)과 외부정렬(보조기억장치)

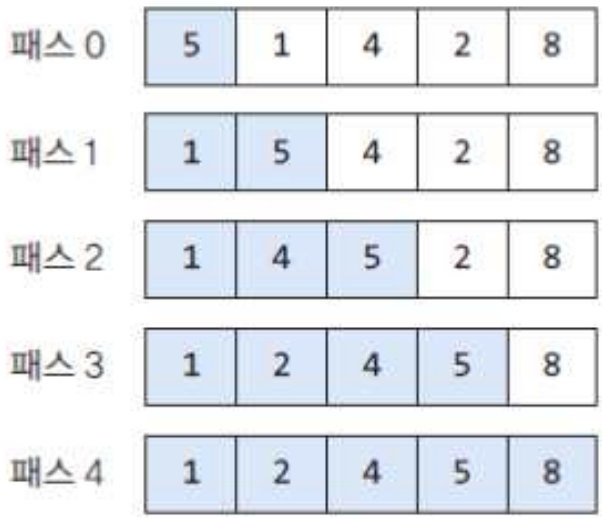
2. 비교 정렬 알고리즘

① 버블 정렬(Bubble Sort)
- 이웃하는 숫자를 비교하여(오름차순의 경우) 큰 수를 오른쪽으로 이동
- 시간 복잡도: 평균 경우 O(N^2), 최악의 경우O(N^2)

[버블 정렬 알고리즘의 예(패스 1의 경우)]

② 삽입 정렬(Insertion Sort)

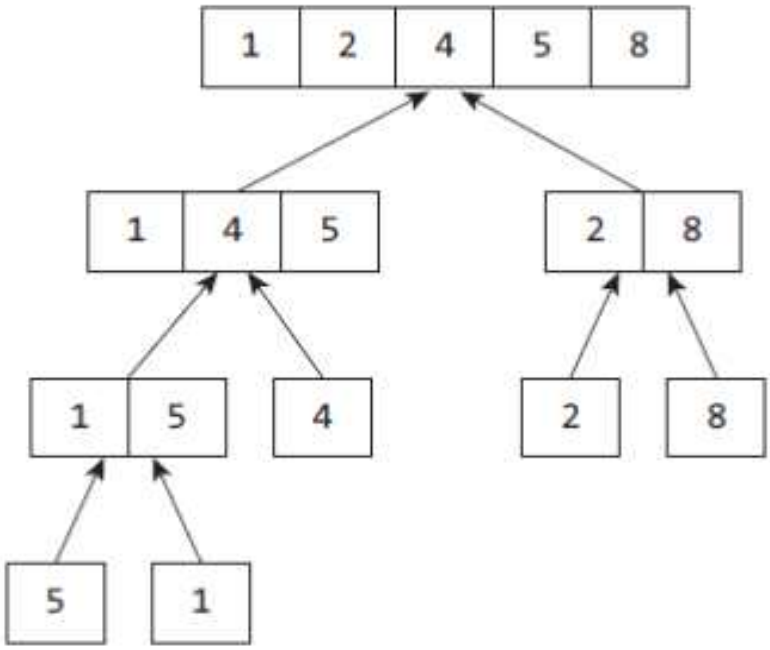
- 정렬이 된 왼쪽 부분과 정렬이 안 된 오른쪽 부분으로 나누고 오른쪽 부분의 맨 왼쪽 요소를 정렬된 부분의 적절한 위치에 삽입하는 방식
- 시간 복잡도: 평균 경우 O(N^2), 최악의 경우 O(N^2)



[삽입 정렬 알고리즘의 개념]

③ 합병 정렬(Merge Sort)

- 정렬할 데이터 수가 N일 때 이것을 $\frac{1}{2}$N 으로 분할하여 정렬한 후,다시 ¼N 으로 된 파일 을 비교 정렬을,… ,파일 크기가 2일 때까지 반복하는 방식
- 시간 복잡도: 평균 경우 O(N logN), 최악의 경우 O(N logN)

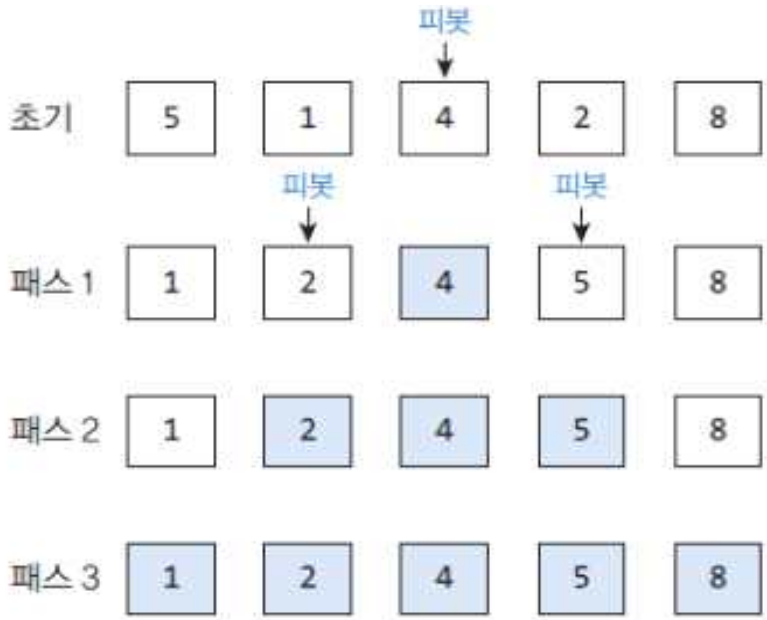


[합병 정렬 알고리즘에서 합병 과정]

- 피봇(Pivot)보다 더 큰 요소는 오른편으로,피봇보다 작은 요소는 왼편으로 위치하도록 분할하고,
- 피봇을 그 사이에 놓는 방식을 반복적으로 수행하는,
- 분할 정복 알고리즘(Divide and Conquer Algorithm)

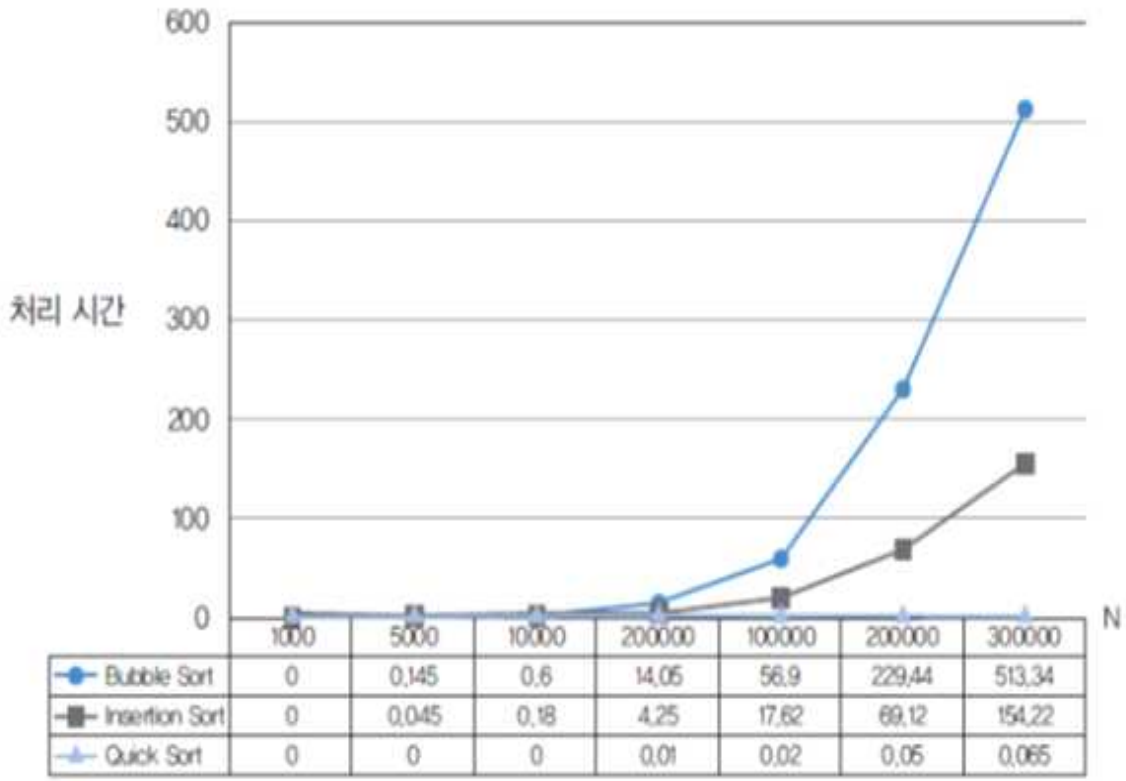
④ 퀵 정렬(Quick Sort)

- 피봇(Pivot)이라 불리는 요소를 중심으로 피봇보다 작은 요소는 왼편으로, 피봇보다 큰 요소는 오른편으로 위치하도록 분할하고 피봇을 그사이에 놓는 방법을 반복하여 정렬하는 방식(Divide and Conquer 알고리즘) 이다.
- 시간 복잡도: 평균 경우 O(N logN), 최악의 경우 O(N^2), 일반적으로 지금까지 알려진 알고리즘 중 평균적으로 가장 빠른 고속 알고리즘



[퀵 정렬 알고리즘의 개념]

* 버블 정렬, 삽입 정렬 및 퀵 정렬의 처리시간 비교



	1000	5000	10000	200000	100000	200000	300000
Bubble Sort	0	0.145	0.6	14.05	56.9	229.44	513.34
Insertion Sort	0	0.045	0.18	4.25	17.62	69.12	154.22
Quick Sort	0	0	0	0.01	0.02	0.05	0.065

학습내용2 : 탐색 알고리즘과 문제 해결방식

- 대형 문고에서 원하는 책을 빠르게 선택하기 위해서는 탐색하기 쉽게 배열되어 있다는 점을 강조

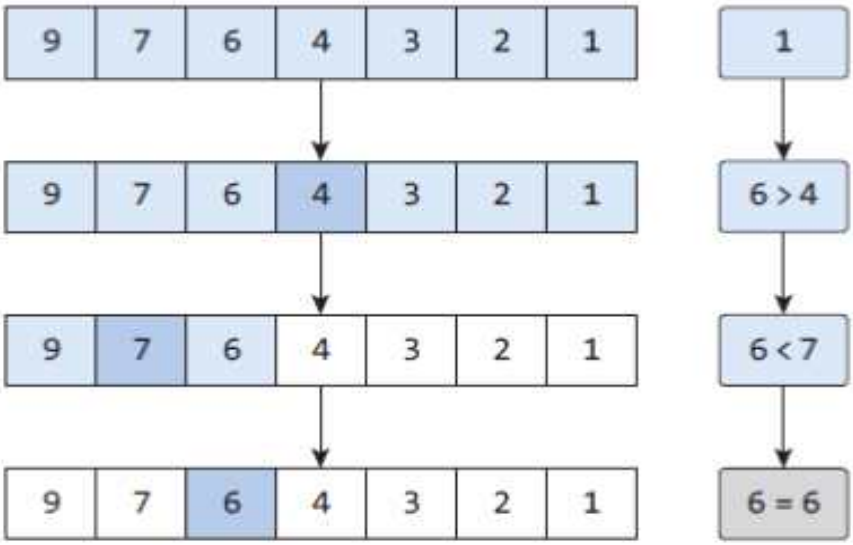
1. 탐색(Searching) 알고리즘

1) 탐색(=검색)이란?
- 수많은 데이터 중 어떤 조건을 만족시키는 데이터를 찾아내는 과정
- 탐색 알고리즘은 정렬 알고리즘과 관련성이 높다.
- 주어진 상황과 가정에 따라(=정렬 여부에 따라) 적용할 탐색 알고리즘이 달라짐
- 예) 웹 정보검색, 데이터베이스의 SQL 질의어 등

2. 탐색 알고리즘의 종류

① 순차 탐색(Sequential Search)
- 임의의 데이터 모임에서 원하는 데이터를 찾기 위해 처음부터 하나씩 비교해 나가는 방식
- 데이터가 정렬되지 않는 경우에 사용
- 시간 복잡도 : 평균 경우 O(N), 최악의 경우 O(N)

② 이진 탐색(Binary Search)
- 원래 데이터가 순서에 따라 정렬되어 있는 경우에 사용할 수 있는 알고리즘
- 정렬된 데이터의 중간을 중심으로 반으로 나누어 중앙 위치의 데이터를 비교, 나누어진 데이터를 다시 반으로 나누어 비교하는 방식
- 시간복잡도 :평균 경우 O(logN), 최악의 경우 O(logN)



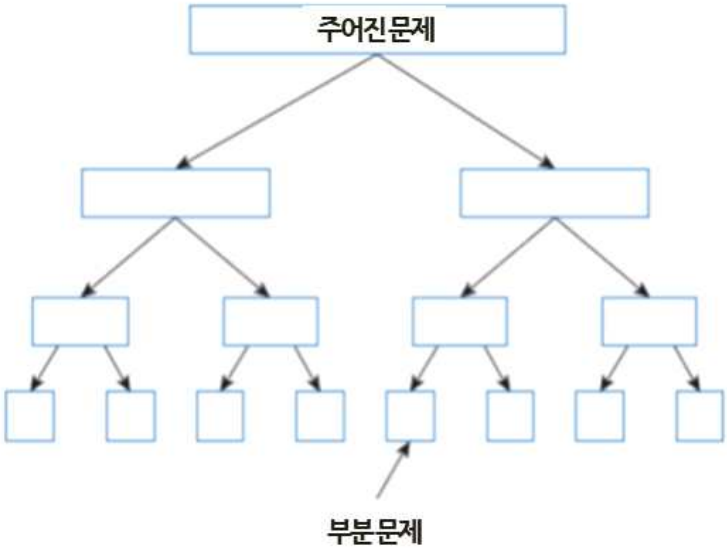
[이진 탐색 알고리즘의 개념]

3. 문제해결 방식

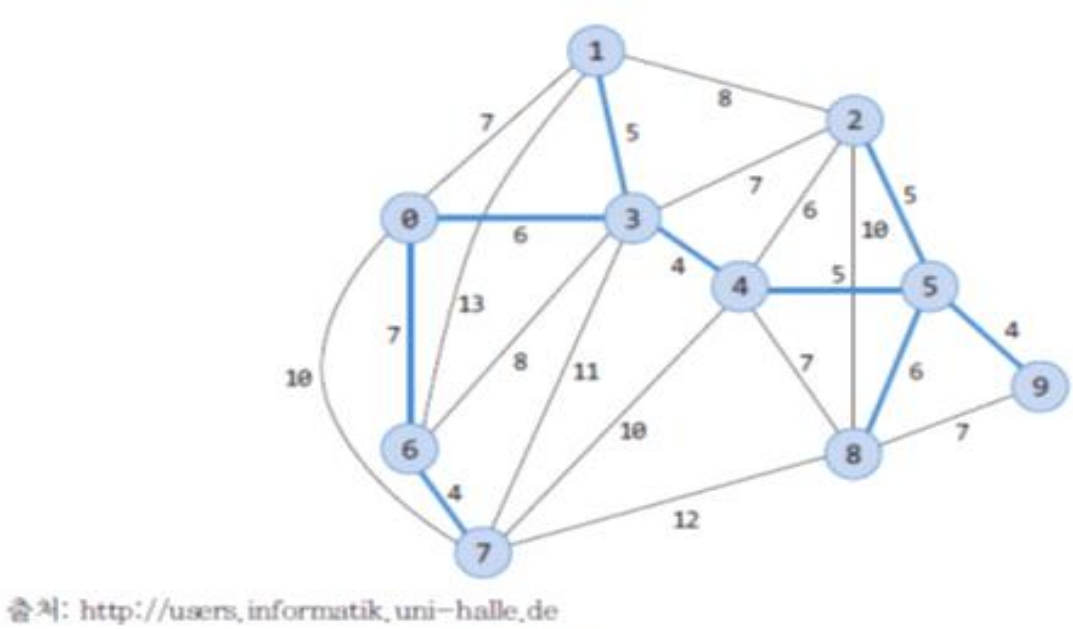
1) 다항식 시간 복잡도
- P(Polynomial) 문제
- 예) 시간 복잡도: O(logN), O(N), O(N logN), O(N^2), O(N^3), O(N^k)

2) NP-완전(NP-Complete) 문제
① P 문제보다 큰 시간 복잡도를 가진 알고리즘으로 제한 점을 주어 P문제로 변형해야 해결
② 대표적인 것이 시간 복잡도가 지수 시간을 가지는 경우: 예를 들면 O(2^n)
③ 어느 하나의 NP-완전 문제에 대하여 다항식 시간 복잡도의 알고리즘을 찾으면 모든 다른 NP-완전 문제도 다항식 시간에 해를 찾을 수 있음(=컴퓨터로 해결 할 수 있는 문제화)
* 분할 정복(Divide-and-Conquer) 알고리즘
- 주어진 문제의 입력을 분할하여 문제를 해결(정복)하는 방식으로
- 분할된 입력(Subproblem)에 대해서 동일한 알고리즘을 적용해 해를 계산하며,
- 이들의 해를 취합하여 원래의 문제의 해를 구하는 방식
- 이용: 퀵 정렬

[분할 정복 알고리즘의 개념]



* 그리디(Greedy) 알고리즘
- 최적화(Optimization) 문제에서 해를 찾을 때 항상 '욕심 내어' 최소값 또는 최대값을 찾아가는 알고리즘(가능한 해들 중에서 가장 좋은 해를 찾는문제)
- 탐욕,욕심쟁이 알고리즘이라 불림
예) 최소 스패닝 트리(Minimum Spanning Tree)



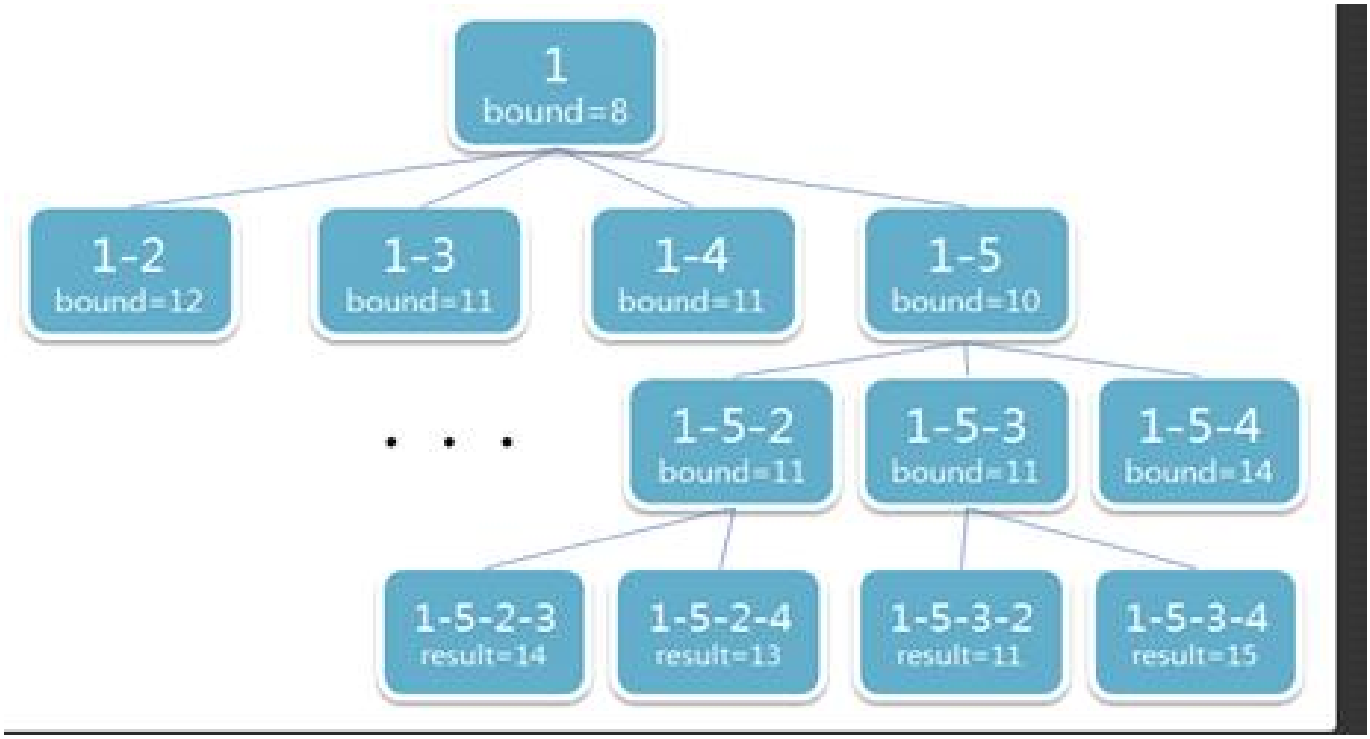
[최소 스패닝 트리의 예]

* 동적 계획(Dynamic Programming) 알고리즘
- 최적화 문제해결 방법으로 입력 크기가 작은 부분문제들을 모두 해결한 후에 그해를 이용해서 점차적으로 큰 문제를 해결하는 방식
(예, 피보나치수열 $A_{n+2} = A_{nn} + A_{n+1}$)

* 근사(Approximation)
- NP-완전 문제를 해결하기 위하여 최적 해에 가까운 근사 해를 구하는 방식

* 백트래킹(Backtracking) 알고리즘
- 해를 찾는 도중에 '막히면' (즉, 해가 아니면) 되돌아가서 다시 해를 찾아가는 방식으로 최적화 문제와 결정 문제를 해결하는 기법(예,8 퀸 문제)

* 분기 한정(Branch-and=Bound) 알고리즘
- 백트래킹의 단점을 보완하는 탐색 기법으로,
- 문제의 조건에 따라 해를 깊이 우선 탐색할 때 특정한 값(한정 값)에 따라 가지치기를 하는 방식



【학습정리】

1. 버블 정렬(Bubble Sort)은 이웃하는 숫자를 비교하여(오름차순의 경우) 큰 수를 오른쪽으로 이동시켜나가는 알고리즘이다.

2. 삽입 정렬(Insertion Sort)은 정렬이 된 왼쪽 부분과 정렬이 안 된 오른쪽 부분으로 나누고 오른쪽 부분의 맨 왼쪽 요소를 정렬된 부분의 적절한 위치에 삽입하는 정렬 알고리즘이다.

3. 순차탐색은 임의의 데이터 모임에서 원하는 데이터를 찾기 위해 처음부터 하나씩 비교해 나가는 탐색 방법이다.