파이썬 심화

정렬

정렬

정렬

- 기본정렬
  - 선택정렬
  - 삽입정렬
  - 버블정렬
- 고급정렬
  - 힙정렬
  - 합병정렬
  - 퀵정렬

기본 정렬

선택정렬

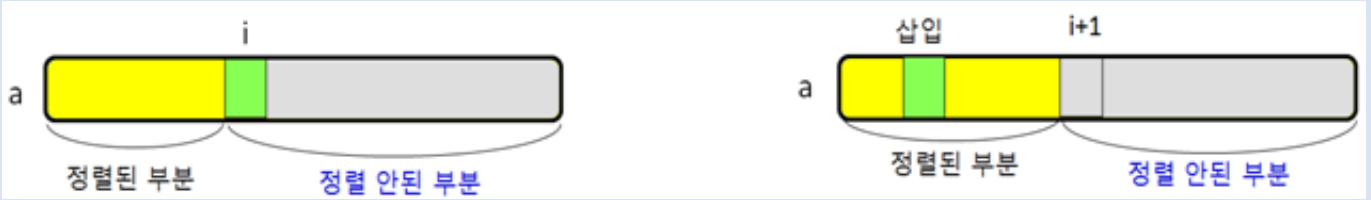
- 선택정렬(Selection Sort)은 배열에서 아직 정렬되지 않은 부분의 원소들 중에서 최솟값을 '선택'하여 정렬된 부분의 바로 오른쪽 원소와 교환하는 정렬알고리즘

선택정렬의 특징

- 입력에 민감하지 않음(Input Insensitive)
  - 항상 $O(N^2)$ 수행시간이 소요
- 최솟값을 찾은 후 원소를 교환하는 횟수가 N-1
  - 이는 정렬알고리즘들 중에서 가장 작은 최악경우 교환 횟수
- 하지만 선택정렬은 효율성 측면에서 뒤떨어지므로 거의 활용되지 않음

삽입정렬

- 삽입정렬(Insertion Sort)은 배열이 정렬된 부분과 정렬되지 않은 부분으로 나뉘며, 정렬 안된 부분의 가장 왼쪽 원소를 정렬된 부분에 '삽입'하는 방식의 정렬알고리즘



삽입정렬의 특징

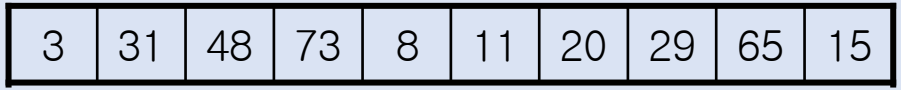
- 입력에 민감함(Input Insensitive)
  - 최선의 경우 O(N) 수행시간이 소요
  - 거의 정렬이 이루어진 입력인 경우 유용
  - 정렬이 이미 이루어진 리스트에 데이터가 추가되는 경우 유용
- 고급정렬의 부분 소규모 정렬에 이용
  - 입력크기가 작은 경우에도 좋은 성능
  - 간단한 알고리즘

버블정렬

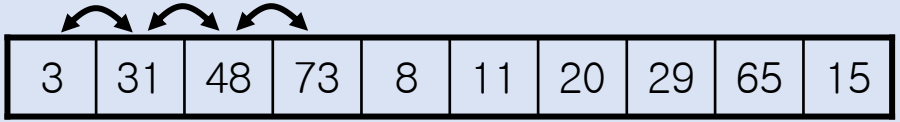
- 선택정렬과 마찬가지로 가장 큰 원소를 오른쪽으로 옮기는 작업을 반복
  - 왼쪽부터 이웃한 수와 비교한다.
  - 이웃한 수가 작은 경우 두 수를 교환한다.
  - 마지막까지 작업이 마쳐지면 n을 줄여 반복한다.

버블정렬의 작동방식

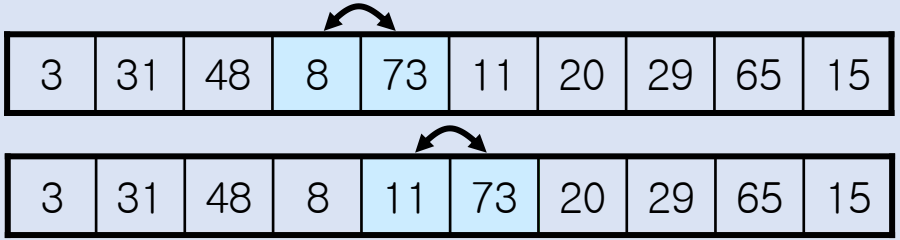
정렬할 배열이 주어진다



왼쪽부터 시작해 이웃한 쌍들을 비교해 나간다



순서대로 되어 있지 않으면 자리 바꾼다



버블정렬의 특징

- 입력에 민감함(Input Insensitive)
  - 최선의 경우 O(N) 수행시간이 소요
  - 데이터 변경 감시가 있는 경우에 한함
- 데이터 이동속도가 느려서 효율이 떨어짐
  - 새로 데이터가 추가된 경우 이동거리가 멀면 여러 번 반복

분할 정복

분할정복 알고리즘

- 주어진 문제의 입력을 분할하여 문제를 해결(정복)하는 방식의 알고리즘
  - 분할한 입력에 대하여 동일한 알고리즘을 적용하여 해를 계산
  - 이들의 해를 취합하여 원래 문제의 해를 얻음

- 부분문제와 부분해
  - 분할된 입력에 대한 문제를 부분문제 (subproblem)
  - 부분문제의 해를 부분해
  - 부분문제는 더 이상 분할할 수 없을 때가지 계속 분할

분할 정복 알고리즘의 분류

- 분할 정복 알고리즘은 분할되는 부분문제의 수와 부분문제의 크기에 따라서 다음과 같이 분류
- 문제가 a개로 분할되고, 부분문제의 크기가 1/b로 감소하는 알고리즘
  - a=b=2인 경우: 합병 정렬, 최근접 점의 쌍 찾기, 공제선 문제
  - a=3, b=2인 경우: 큰 정수의 곱셈
  - a=4, b=2인 경우: 큰 정수의 곱셈
  - a=7, b=2인 경우: 스트라센(Strassen)의 행렬 곱셈 알고리즘

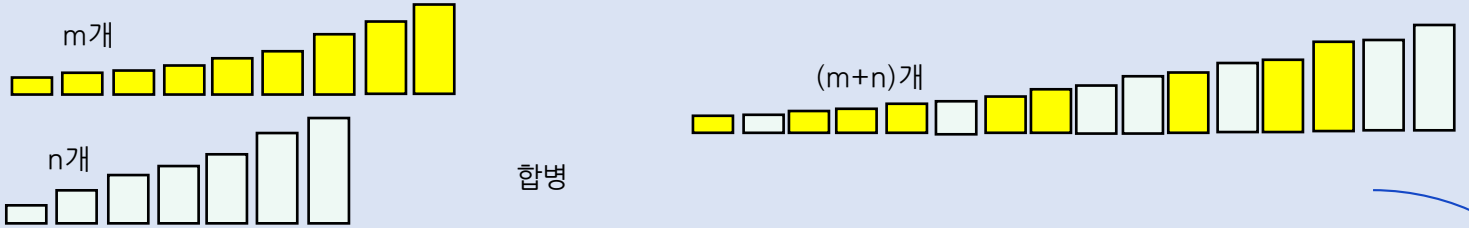
고급 정렬

합병 정렬

- 합병 정렬은 입력의 크기가 ½로 줄고, 문제가 2개로 나누어지는 대표적인 분할 정복 알고리즘을 적용하는 정렬 방식
  - 1 7 2 4 6 5 3 9 → 1 7 2 4 문제와 6 5 3 9 문제로 나눔
  - 적절한 정렬과정을 통하여 각각의 문제 해결
- 합병(merge) 과정을 통해서 정렬이 완료된 두개의 결과를 이용하여 한 개의 정렬된 결과 도출
  - 1 2 4 7 자료와 3 5 6 9 문제를 병합 → 1 2 3 4 5 6 7 9

합병 시간 복잡도

- 두개의 자료(m개의 정렬된 자료, n개의 정렬된 자료)를 병합하는 과정
  - 가장 작은 값끼리 비교하여 작은 값을 선택해서 데이터 저장
  - 위과정을 최대 m+n-1 번 반복하면 정렬된 데이터 얻음
  - O(m+n)



시간 복잡도

- 입력이 n인 데이터를 합병정렬로 정렬하는 시간을 T(n)이라고 했을때,
  - 분할된 입력을 정렬하는 시간은 T(n/2)
  - n/2로 분할된 두 데이터를 합병하는데 걸리는 시간은 O(n)
  - T(n) = 2T(n/2) + O(n)
  - 결과 : T(n) = O(n log n)

합병정렬의 특징

- 병렬 처리에 적용 가능 → 멀티 프로세스나 분산처리에 유용
- 외부 정렬로 사용 가능 → 대규모 데이터 정렬
- 자료구조가 연결리스트인 경우에도 사용 가능
- 정적 정렬(static sort) 가능
- 공간 복잡도가 O(n)으로 다른 고급정렬 O(1)에 비해 추가 공간 필요

퀵 정렬

- 기준 원소를 기준으로 데이터를 두 부분으로 나눈 다음 각각의 부분을 정렬하는 분할 정복 알고리즘을 적용하는 정렬
  - 1 7 2 4 9 5 3 6 → 6을 기준으로 1 2 4 5 3 [6] 7 9 로 나눔
  - 각각의 부분을 정렬 → 1 2 3 4 5 [6] 7 9 (정렬 완료)

시간 복잡도

- 입력이 n인 데이터를 퀵 정렬로 정렬하는 시간을 T(n)이라고 했을때,
  - 기준원소를 기준으로 두부분으로 나누는 시간복잡도가 O(n)
  - 분할된 영역의 평균 크기가 n/2라 할 때, 분할된 입력을 정렬하는 시간은 T(n/2)
  - $T(n) = 2T(n/2) + O(n)$
  - 결과 : $T(n) = O(n \log n)$
  - 분할된 영역이 한쪽으로 치우치게 되면, $T(n) = O(n^2)$ 으로 평가됨

퀵 정렬의 특징

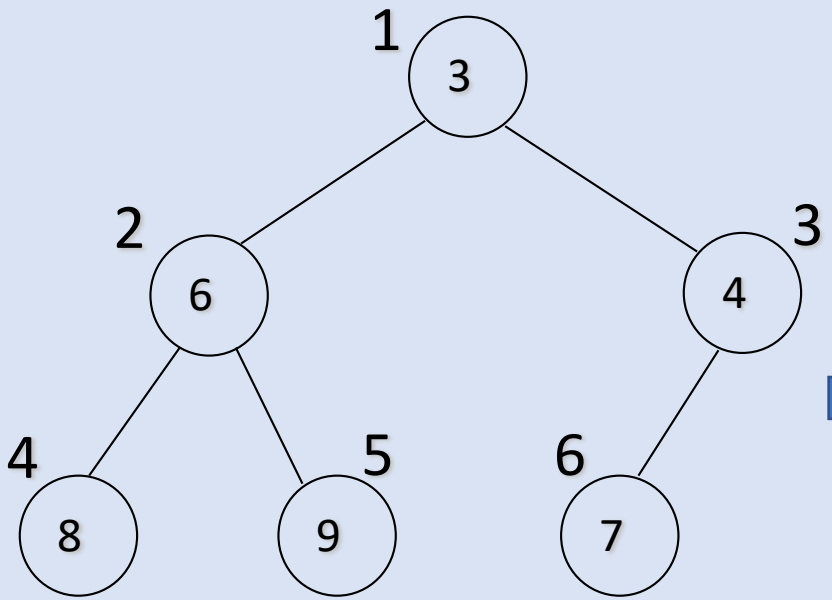
- 캐쉬 효율과 일부 비재귀 처리로 평균 속도가 가장 빠름
- 기준원소 선택에 따라 정렬 효율이 달라짐 (개선방법 3-pivot 등)

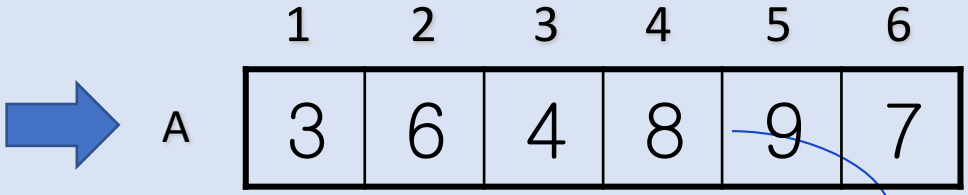
힙 정렬

- 힙(Heap)자료구조를 이용한 정렬 방법
  - 데이터에 대한 힙 만들기 (Build Heap)
  - 최소(또는 최대)값을 제거하여 데이터 모으기
- 힙(Heap) 자료구조
  - 꽉찬 이진 트리(Binary Tree)
  - 각 노드에 값이 저장
  - 부모의 노드값이 자식의 노드값보다 항상 작은 값

힙의 리스트 표현법

- 꽉찬 이진 트리로 각각의 노드값이 리스트 값으로 표현됨





시간 복잡도

- 입력이 n인 데이터를 힙 정렬로 정렬하는 시간을 T(n)이라고 했을때,
  - 힙을 만드는 시간복잡도 O(n)
  - 최소(최대)값을 가져오는 시간 복잡도 O(log* n)
  - 최소(최대)값을 가져오는 횟수 n-1
  - 결과 : T(n) = O(n log* n)

힙 정렬의 특징

- 이론적으로 가장 빠른 시간복잡도 O(n log* n)
- 캐시 효율이 나쁨

실습

합병 정렬 구현

- N개의 데이터를 랜덤하게 생성하고, 합병 정렬을 구현, 성능을 평가한다.

퀵 정렬 구현

- N개의 데이터를 생성하고, 퀵 정렬을 구현, 성능을 평가한다.

힙 정렬 구현

- N개의 데이터를 랜덤하게 생성하고, 힙 정렬을 구현하고, 성능을 평가한다.

히스트그램에서 가장 넓은 직사각형 넓이 구하기

- 아래 그림과 같은 형태의 히스토그램이 있을 때, 가로축의 히스토그램의 길이를 1로, 세로축의 히스토그램 크기는 자연수일 때, 이 히스토그램에서 만들 수 있는 최대 크기의 직사각형 넓이를 구한다.
  - [3, 2, 3. 4, 2, 3, 4, 1]의 경우 최대 직사각형 넓이는 14
  - 참고 : https://www.acmicpc.net/problem/1725