정렬 알고리즘 정리

선택 정렬(select sort)

두가지 방법이 있음.

- 최대값을 선택
- 최솟값을 선택

최솟값을 선택하는 방법에대해서 설명하겠습니다.

1. 배열의 원소중 최솟값을 찾아서 선택한다.
2. 첫번째 원소와 자리를 교환한다.
3. 1회 수행하고 나면 두번째, 세번째 … N번째의 원소와 교환할 최솟값을 찾고 교환한다.
4. 배열의 끝까지 탐색하면 정렬을 마친다.

예시

1. 29 10 14 37 13
2. ~~10~~ 29 14 37 13
3. ~~10 13~~ 14 37 29
4. ~~10 13 14~~ 37 29
5. ~~10 13 14 29 37~~

최솟값을 찾는 시간복잡도 - O(n)

swap을 하는 시간복잡도 - O(n)

따라서 총 시간복잡도는 O(n^2)

배열의 상태가 이미 정렬된 경우 → 최선의 시간복잡도 = O(n)

선택 정렬은 불안정 정렬이다.(동일한 값에 대한 기존의 순서가 유지를 보장 못함)

버블 정렬(bubble sort)

1. 서로 인접한 원소들끼리 비교하여 정렬 순서에 맞지 않으면 교환한다. → 최대 원소를 가장 뒤로보냄
2. 최대 원소를 제외한 나머지 원소들을 같은방법으로 교환한다.
3. 배열의 끝까지 반복

예시

1. 29 10 14 37 13
2. 10 29 14 37 13
3. 10 14 29 37 13
4. 10 14 29 37 13
5. 10 14 29 13 ~~37~~

계속 반복…

시간복잡도 - O(n^2)

버블 정렬은 안정 정렬이다(동일한 값에 대한 기존의 순서가 유지됨)

삽입 정렬 (insert sort)

- 두번째 원소부터 시작하며 그 앞 원소들과 비교하여 삽입할 위치를 정한 후 자료를 right shift(오른쪽으로 밀기)한 후 해당 원소를 삽입한다.
- 3번째, 4번째 …. N번째까지 반복

예시

1. ~~29~~ 10 14 37 13 ← 10 복사
2. 29 29 14 37 13 ← 10보다 큰 29를 오른쪽으로 shift
3. ~~10 29~~ 14 37 13 ← 10 삽입, 14를 복사

계속 반복…

시간복잡도 - O(n^2)

배열의 상태가 이미 정렬된 경우 → 최선의 시간복잡도 = O(n)

병합 정렬(merge sort)

재귀적으로 정렬을 하는 방법.

1. 정렬할 대상을 반을 나눈다.
2. 나눈 전반부와 후반부를 각각 독립적으로 정렬한다. 부분 배열의 크기가 충분하지 않으면 다시 1번 호출
3. 정렬된 두 부분을 합쳐서 정렬된 배열을 얻는다.

예시

31 3 65 73 8 11 20 29

…

31 3 65 73 | 8 11 20 29

31 3 | 65 73 | 8 11 | 20 29 ← 최대한 반으로 나눈다(재귀적으로 계속 반으로 나눔)

3 31 | 65 73 | 8 11 | 20 29 ← 정렬

3 31 65 73 | 8 11 20 29 ← 합병과 정렬

3 8 11 20 29 31 65 73 ← 끝

병합 정렬은 재귀 알고리즘이며 점화식으로 표현하면 아래와 같음

- 수행 시간 = 두 부분배열을 각각 정렬하는 시간 + 병합시간
  - T(n) = 2T(n/2) + ~~Θ~~(n)
- 마스터 정리를 이용하여 점근적 표기를 구할 수 있음
  - T(n) = ~~Θ~~(n log n)

시간복잡도 - O(n log n)

병합 정렬은 안정 정렬이다

퀵 정렬(quick sort)

1. 정렬할 배열에서 기준 원소를 하나 고른다.
2. 이 기준 원소를 중심으로 더 작거나 같은 원소는 왼쪽으로, 큰 원소는 오른쪽으로 재배치해서 배열을 분할한다.
3. 이렇게 분할된 왼쪽, 오른쪽 부분 배열을 각각 독립적으로 정렬한다.

예시

31 8 48 73 11 3 20 29 65 15 ← 가장 끝에 있는 수를 기준(pivot)으로 삼기(다른 수를 pivot으로 해도됨)

8 11 3 15 31 48 20 29 65 73 ← 기준 이하의 수를 왼쪽, 기준보다 큰 경우는 오른쪽으로 재배치(partition 알고리즘)

3 8 11 15 20 29 31 48 65 73 ← 왼쪽, 오른쪽을 독립적으로 정렬

최선, 평균 시간복잡도 - O(n log n)

최악 시간복잡도 - O(n^2) ← 각 partition 진행 후 pivot이 해당 배열의 가장 크거나 작은 경우

퀵 정렬은 불안정 정렬이다.

TMI

Java에서 Array.sort() 메소드는 해당 배열의 형에 따라서 알고리즘이 다르다.

일반 배열(int형)일 경우 Dual Pivot Quick sort를 사용한다. ← 위에서 설명한 알고리즘은 pivot이 한개이다.

객체 배열(Object형)일 경우 Tim sort를 사용한다. ← 병합 정렬과 선택 정렬을 이용한 알고리즘