

# #3. 정렬과 탐색의 하계

나정휘

<https://justicehui.github.io/>

# 목차

연산 횟수의 하계(lower bound)

최댓값과 최솟값 찾기

두 번째로 큰 값 찾기

# 연산 횟수의 하계

# 연산 횟수의 하계

## 알고리즘의 시간 복잡도

- 시간 복잡도를 계산하는 방법
    - 주어진 문제에 대한 필수적인 **기본 연산**을 정의한 뒤
    - 알고리즘이 기본 연산을 몇 번 수행하는지 세면 됨
  - 배열에서 어떤 키 K를 찾는 문제 → 배열의 원소  $A[i]$ 와 K를 비교하는 연산
  - 배열의 원소를 정렬하는 문제 → 배열의 두 원소  $A[i]$ 와  $A[j]$ 의 크기를 비교하는 연산
  - 원소가 실수인 두 행렬의 곱 → 두 실수의 덧셈 또는 곱셈
- 알고리즘의 최적성
    - 더 이상 개선할 여지가 없을 만큼 최적화가 되었는지 어떻게 판단할 수 있을까?
    - 가능한 모든 알고리즘을 분석해야 할까?

# 연산 횟수의 하계

## 알고리즘의 시간 복잡도

- 알고리즘의 최적성
  - 더 이상 개선할 여지가 없을 만큼 최적화가 되었는지 어떻게 판단할 수 있을까?
  - 가능한 모든 알고리즘을 분석해야 할까?
  - 문제를 풀기 위해서 필수적인 기본 연산의 최소 횟수  $F(n)$ 을 구한 뒤
  - 주어진 알고리즘에서 기본 연산의 시행 횟수가  $F(n)$ 과 같은지 비교하면 됨
- 예시
  - 두  $N \times N$  크기의 행렬의 곱셈
    - $2N^2$ 개의 원소를 모두 확인해야 하므로  $O(N^2)$ 보다 더 적은 연산으로 불가능
  - 배열에서 최댓값 찾기
    - 모든 원소가 서로 다르다고 가정하자.
    - $N$ 개의 원소 중  $N-1$ 개의 원소는 최댓값이 아니라는 것을 확인해야 최댓값을 확정지을 수 있음
    - 따라서  $F(N) = N-1$ 이 비교 연산 횟수의 하계

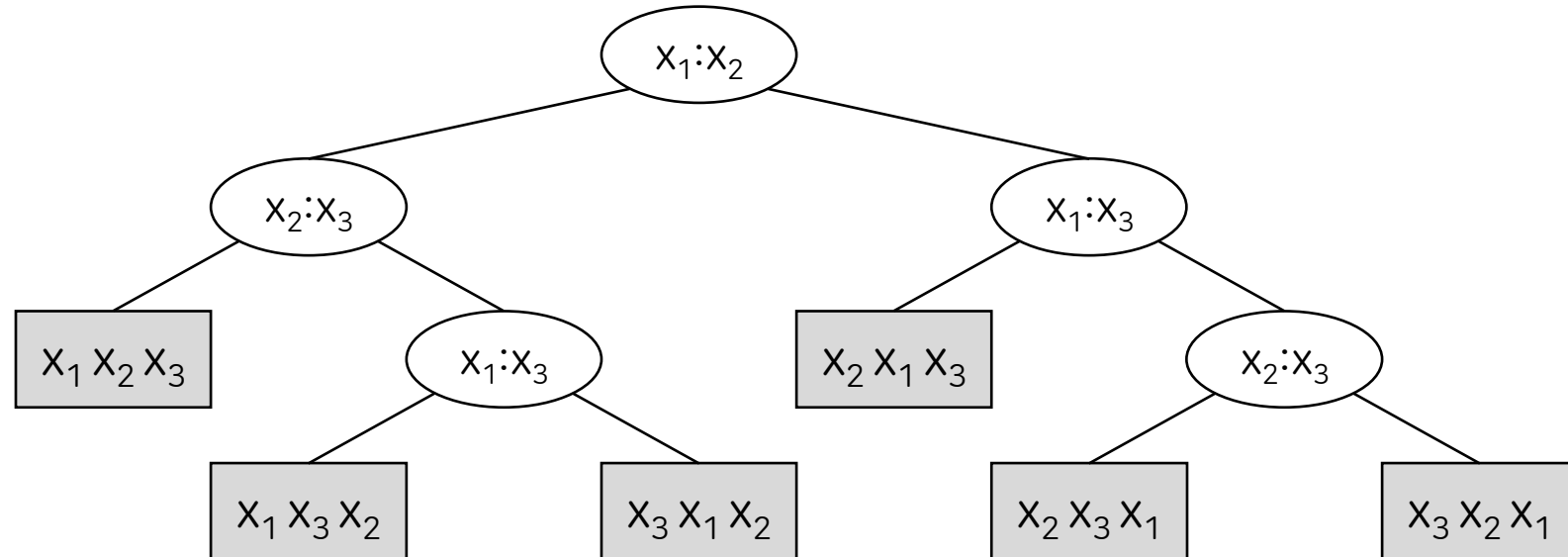
# 연산 횟수의 하계

## 비교 기반 정렬의 하계

- 비교 기반 정렬에서 worst case의 시간 복잡도 하계는  $\Theta(n \log n)$ 이다.
- decision tree를 생각하자.
  - 비교 기반 정렬을 이용해 서로 다른  $n$ 개의 원소  $x_1, x_2, \dots, x_n$ 을 정렬하는 과정은 decision tree로 표현 가능
  - $n!$ 개의 리프 정점이 있고, 모든 내부 정점의 자식이 2개인 이진 트리
  - 비교 연산 횟수 = 루트에서 리프 정점까지의 거리
- $n = 3$ 일 때의 삽입 정렬을 생각해 보자.



```
for(int i=2; i<=n; i++){  
    for(int j=i; j>1; j--){  
        if(a[j] > a[j+1]) swap(a[j], a[j+1]);  
        else break;  
    }  
}
```



# 연산 횟수의 하계

## 비교 기반 정렬의 하계

- 비교 기반 정렬에서 worst case의 시간 복잡도 하계는  $\Theta(n \log n)$ 이다.
- decision tree를 생각하자.
  - 비교 기반 정렬을 이용해 서로 다른  $n$ 개의 원소  $x_1, x_2, \dots, x_n$ 을 정렬하는 과정은 decision tree로 표현 가능
  - $n!$ 개의 리프 정점이 있고, 모든 내부 정점의 자식이 2개인 이진 트리
  - 비교 연산 횟수 = 루트에서 리프 정점까지의 거리
- 비교 기반 정렬 알고리즘의 worse case = decision tree의 높이
  - 트리를 어떻게 만들더라도 높이가  $\Omega(n \log n)$ 임을 증명하면 됨

# 연산 횟수의 하계

## 비교 기반 정렬의 하계

- 비교 기반 정렬에서 worst case의 시간 복잡도 하계는  $\Theta(n \log n)$ 이다.
- 비교 기반 정렬 알고리즘의 worst case = decision tree의 높이
  - 트리를 어떻게 만들더라도 높이가  $\Omega(n \log n)$ 임을 증명하면 됨
- 증명
  - 이진 트리의 높이를  $h$ , 리프의 개수를  $m$ 이라고 하면  $m \leq 2^h$
  - 양변에 로그를 취하면  $h \geq \text{ceil}(\log_2 m)$
  - $\rightarrow n$ 개의 원소를 정렬하는 모든 비교 기반 정렬은 최악의 경우에  $\text{ceil}(\log_2 n!)$ 번 이상의 비교를 수행해야 함
  - 또한  $\log_2 n! \in \Theta(n \log n)$  이므로, 결국 비교 기반 정렬의 worst case의 하계는  $\Theta(n \log n)$ .

# 연산 횟수의 하계

## 인접한 원소를 교환하는 정렬의 하계

- 인접한 원소를 교환하는 정렬에서 average case의 시간 복잡도 하계는  $\Theta(n^2)$ 이다.
- 평균의 경우도 비슷한 방법으로 증명할 수 있다!
- inversion
  - $a < b$  이면서  $x_a > x_b$ 인 순서쌍  $(a, b)$ 를 inversion이라고 하자.
  - inversion의 개수가 0이면 정렬된 배열
  - $n$ 개의 원소의 값이 모두 서로 다르다면, 인접한 두 원소를 교환할 때 inversion 개수는 1만큼 증가 또는 감소함
  - 따라서 인접한 원소를 교환하는 정렬 알고리즘은 비교 연산을 최소한 inversion의 개수 이상 수행해야 함

# 연산 횟수의 하계

## 인접한 원소를 교환하는 정렬의 하계

- 인접한 원소를 교환하는 정렬에서 average case의 시간 복잡도 하계는  $\Theta(n^2)$ 이다.
- 길이가  $n$ 이고 모든 원소의 값이 다른 순열의 inversion 개수의 기댓값은?
  - 앞에서 배운 삽입 정렬의 평균 시간 복잡도를 구하는 것과 같은 방법으로 계산할 수 있음
  - $n * (n-1) / 4$
- 따라서 인접한 원소를 교환하는 정렬에서 average case의 하계는  $\Theta(n^2)$ .

# 연산 횟수의 하계

## 상대자 논증 (adversary argument)

- 업/다운 게임
  - 철수는 1 이상  $N$  이하인 자연수  $X$ 를 하나 선택한다.
  - 영희는 철수가 생각한  $X$ 를 맞혀야 한다.
  - 영희가 어떤 수  $Y$ 를 말하면, 철수는 아래와 같은 대답을 한다.
    - $X > Y$ 라면 Up,  $X < Y$ 라면 Down,  $X = Y$ 라면 Accept
  - 영희의 질문 횟수를 최소화 시키자.
- 철수가 악의를 품으면?
  - 철수는  $X$ 를 미리 생각하지 않음
  - 영희의 질문에 맞게, **모순이 발생하지 않는 선에서** 영희의 질문 횟수가 최대가 되도록 대답함
  - 예시 ( $N = 100$ )
    - 영희의 질문: 10 → 철수의 대답: Up
    - 영희의 질문: 90 → 철수의 대답: Down

# 연산 횟수의 하계

## 상대자 논증 - 예시 문제

- 오름차순으로 정렬된 배열에서 값이  $K$ 인 원소가 존재하는지 판별
  - 기본 연산: 배열의 한 원소와  $K$ 를 비교하는 것
  - 업/다운 게임과 같은 상황
- 상대자의 전략
  - 현재 정답이 존재할 수 있는 구간을  $[L, R]$ 이라고 하자. ( $n = R - L + 1$ )
  - 알고리즘이 어떤 지점  $p$ 를 찍으면,  $[L, p)$ 와  $(p, R]$  중 더 구간의 길이가 긴 쪽으로 유도
  - 상대자는 알고리즘이 어떤 위치를 찍더라도, 항상 정답이 존재할 수 있는 구간을  $\text{floor}(n/2)$  이상으로 유지 가능
- 따라서 어떤 알고리즘도 최악의 경우에 비교 연산을  $\Theta(\log n)$ 보다 적게 수행할 수 없음

최댓값과 최솟값 찾기

# 최댓값과 최솟값 찾기

## 문제 설명

- 입력: 서로 다른  $n$ 개의 정수로 구성된 배열
- 할 수 있는 연산: 원소의 복사/이동, 배열의 두 원소의 대소 비교
- 목표: 최소한의 비교 연산으로 최대값과 최솟값 찾기
- 가장 간단한 풀이
  - $n-1$ 번의 비교를 통해 최댓값을 찾은 뒤
  - $n-2$ 번의 비교를 통해 최댓값을 제외한  $n-1$ 개의 원소 중 최솟값 찾기
  - 비교 연산 횟수 =  $(n-1) + (n-2) = 2n - 3$
- 비효율적인 이유
  - 최댓값을 찾을 때 수행한 비교 연산의 결과를 최솟값을 찾을 때 사용하지 않음

# 최댓값과 최솟값 찾기

## 풀이

- 목표: 최소한의 비교 연산으로 최대값과 최솟값 찾기
- 풀이
  1. 원소를 둘씩 짝지어서  $\text{floor}(n/2)$ 번의 비교 연산 수행
    - $n$ 이 홀수면 마지막 원소는 (2)와 (3)에서 모두 사용
  2. (1)에서 더 큰 원소  $\text{ceil}(n/2)$ 개 중에 최댓값 찾기
  3. (1)에서 더 작은 원소  $\text{ceil}(n/2)$ 개 중에 최솟값 찾기
- 비교 연산 횟수
  - (1)에서  $\text{floor}(n/2)$ 번
  - (2)와 (3)에서 각각  $\text{ceil}(n/2) - 1$ 번
  - $n$ 이 짝수면  $n/2 + (n/2 - 1) + (n/2 - 1) = 3n/2 - 2$ 번
  - $n$ 이 홀수면  $(n-1)/2 + (n-1)/2 + (n-1)/2 = 3n/2 - 3/2$ 번
- 따라서 총 비교 연산 횟수는  $\text{ceil}(3n/2) - 2$  이하

# 최댓값과 최솟값 찾기

## 상대자의 전략

- 문제를 풀기 위해 필요한 정보
  - 최댓값을 제외한 모든 원소는 비교에서 진 적이 있어야 함
  - 최솟값을 제외한 모든 원소는 비교에서 이긴 적이 있어야 함
  - 총  $2(n - 2)$ 개의 정보를 얻으면 답을 확정할 수 있음
- 배열의 원소마다 아래 4가지 중 하나의 태그를 붙이자
  - N: 아직 비교한 적 없음
  - W: 비교에서 승리한 적 있음 / 패배한 적 없음
  - L: 비교에서 패배한 적 있음 / 승리한 적 없음
  - WL: 비교에서 승리와 패배 모두 함
  - $(n-2)$ 개의 WL 태그, 1개의 W 태그, 1개의 L 태그를 모으면 끝
- 상대자의 목표:  $2(n - 2)$ 개의 정보를 최대한 천천히 주기
  - 비교하는 두 원소의 태그가 모두 N일 때는 어쩔 수 없이 2개의 정보를 줘야 함
  - 다른 모든 경우에 최대 1개의 정보만 제공하도록 해야 함

# 최댓값과 최솟값 찾기

## 상대자의 전략

- 상대자의 목표:  $2(n - 2)$ 개의 정보를 최대한 천천히 주기
  - 비교하는 두 원소의 태그가 모두 N일 때는 어쩔 수 없이 2개의 정보를 줘야 함
  - 다른 모든 경우에 최대 1개의 정보만 제공하도록 해야 함

| 피연산자의 태그 | 상대자의 응답      | 비교 이후 태그 | 제공하는 정보량 |
|----------|--------------|----------|----------|
| N, N     | $x > y$      | W, L     | 2        |
| N, W     | $x < y$      | L, W     | 1        |
| N, L     | $x > y$      | W, L     | 1        |
| N, WL    | $x > y$      | W, WL    | 1        |
| W, W     | $x > y$      | W, WL    | 1        |
| W, L     | $x > y$      | W, L     | 0        |
| W, WL    | $x > y$      | W, WL    | 0        |
| L, L     | $x < y$      | L, WL    | 1        |
| L, WL    | $x < y$      | L, WL    | 0        |
| WL, WL   | 모순이 발생하지 않도록 | WL, WL   | 0        |

# 최댓값과 최솟값 찾기

## 상대자의 전략

- 필요한 비교 횟수 ( $n$ 은 짝수라고 가정)
  - 처음에  $n/2$ 번의 연산을 통해 매번 2개씩, 총  $n$ 개의 정보를 얻을 수 있음
  - 남은  $n-2$ 개의 정보는 비교 1번 당 정보를 1개씩 얻을 수 있음
  - 따라서 최소한  $n/2 + n - 2 = 3n/2 - 2$ 번의 연산 필요
- $n$ 이 홀수라면?
  - 처음에  $(n-1)/2$ 번의 연산을 통해  $n-1$ 개의 정보를 얻음
  - 이후  $n-1$ 번의 비교를 통해 남은  $n-1$ 개의 정보를 얻음
  - 따라서 최소한  $3n/2 - 3/2$ 번의 연산 필요
- 따라서 이 문제의 비교 횟수의 하한은  $\text{ceil}(3n/2) - 2$ .

두 번째로 큰 값 찾기

# 두 번째로 큰 값 찾기

## 문제 설명

- 입력: 서로 다른  $n$ 개의 정수로 구성된 배열
- 할 수 있는 연산: 원소의 복사/이동, 배열의 두 원소의 대소 비교
- 목표: 최소한의 비교 연산으로 두 번째로 큰 값 찾기
- 몇 가지 관찰
  - 두 번째로 큰 값을 찾기 위해서는 무조건 최댓값을 알아야 한다.
    - proof. 두 번째로 큰 값이라는 것을 알기 위해서는 나보다 큰 값(최댓값)이 1개, 작은 값이  $n-2$ 개 있다는 것을 알아야 함
  - 최댓값을 제외한 다른 값에게 패배한 원소는 두 번째로 큰 값이 될 수 없다.
    - proof. 자신보다 큰 원소가 2개 이상 존재함
- 전체적인 알고리즘의 흐름
  - 최댓값은 무조건 구해야 하므로, 일단  $n-1$ 번의 비교를 통해 최댓값을 구함
  - 최댓값에게 직접 패배한 원소가  $k$ 개일 때,  $k-1$ 번의 비교를 통해 최댓값(두 번째로 큰 값)을 구함
  - 전체 비교 횟수 =  $n + k - 2$ ,  $k$ 를 최소화해야 함

# 두 번째로 큰 값 찾기

## 풀이

- 목표: 최소한의 비교 연산으로 두 번째로 큰 값 찾기
- 풀이
  1. 토너먼트 방식으로 값을 비교하면서 최댓값을 구함 / 각 원소는 최대  $\text{ceil}(\log_2 n)$ 개의 라운드에 참가함
  2. (1)에서 최댓값에게 패배한  $\text{ceil}(\log_2 n)$ 개의 원소들 중 최댓값을 구함
- 비교 연산 횟수
  - (1)에서  $n-1$ 번
  - (2)에서  $\text{ceil}(\log_2 n) - 1$ 번
- 따라서 총 비교 연산 횟수는  $n + \text{ceil}(\log_2 n) - 2$ 번

# 두 번째로 큰 값 찾기

## 문제 설명

- 입력: 서로 다른  $n$ 개의 정수로 구성된 배열
- 할 수 있는 연산: 원소의 복사/이동, 배열의 두 원소의 대소 비교
- 목표: 최소한의 비교 연산으로 두 번째로 큰 값 찾기
- 몇 가지 관찰
  - 두 번째로 큰 값을 찾기 위해서는 무조건 최댓값을 알아야 한다.
    - proof. 두 번째로 큰 값이라는 것을 알기 위해서는 나보다 큰 값(최댓값)이 1개, 작은 값이  $n-2$ 개 있다는 것을 알아야 함
  - 최댓값을 제외한 다른 값에게 패배한 원소는 두 번째로 큰 값이 될 수 없다.
    - proof. 자신보다 큰 원소가 2개 이상 존재함
- 전체적인 알고리즘의 흐름
  - 최댓값은 무조건 구해야 하므로, 일단  $n-1$ 번의 비교를 통해 최댓값을 구함
  - 최댓값에게 직접 패배한 원소가  $k$ 개일 때,  $k-1$ 번의 비교를 통해 최댓값(두 번째로 큰 값)을 구함
  - 전체 비교 횟수 =  $n + k - 2$ ,  $k$ 를 최소화해야 함

# 두 번째로 큰 값 찾기

## 상대자의 전략

- 전체적인 알고리즘의 흐름
  - 최댓값은 무조건 구해야 하므로, 일단  $n-1$ 번의 비교를 통해 최댓값을 구함
  - 최댓값에게 직접 패배한 원소가  $k$ 개일 때,  $k-1$ 번의 비교를 통해 최댓값(두 번째로 큰 값)을 구함
  - 전체 비교 횟수 =  $n + k - 2$
- 목표: 최댓값에게 직접 패배한 원소의 개수가 항상  $\text{ceil}(\log_2 n)$  이상이 되도록 하는 전략 찾기
- 원소의 가중치  $w(x)$ 
  - $w(x)$ 를 (현재까지  $x$  보다 작은 것으로 확정된 원소의 수) + 1 라고 정의하자.
  - 단, 만약  $w(x)$ 가 최댓값이 아님이 확정되었다면 0
  - $w(x)$ 는 처음에 1로 초기화된 상태
  - $x$ 와  $y$ 를 비교한 결과가  $x > y$  라면,  $w(x)$ 는  $w(y)$  만큼 증가하고  $w(y)$ 는 0으로 바뀜
  - 즉,  $w(x)$ 의 합은 언제나  $n$ 으로 유지됨
- 목표: 최대한 천천히  $w$ 의 최댓값이  $n$ 이 되도록 만들기

# 두 번째로 큰 값 찾기

## 상대자의 전략

- 원소의 가중치  $w(x)$ 
  - $w(x)$ 는 처음에 1로 초기화된 상태
  - $x$ 와  $y$ 를 비교한 결과가  $x > y$  라면,  $w(x)$ 는  $w(y)$  만큼 증가하고  $w(y)$ 는 0으로 바뀜
  - 즉,  $w(x)$ 의 합은 언제나  $n$ 으로 유지됨
- 목표: 최대한 천천히  $w$ 의 최댓값이  $n$ 이 되도록 만들기
  - $w(x) > w(y)$ 일 때  $x > y$  라고 답하는 것이  $x < y$  라고 답하는 것보다 유리함
  - 또한,  $x > y$ 라고 대답하면  $w$ 의 최댓값은 매번 2배 이하로 증가함
- 따라서 아래와 같이 대답하면  $w$ 의 최댓값이  $n$ 이 되기까지 항상  $\text{ceil}(\log_2 n)$ 번 이상의 비교가 필요함

| 피연산자의 가중치         | 상대자의 응답      | 가중치 수정                                               |
|-------------------|--------------|------------------------------------------------------|
| $w(x) > w(y)$     | $x > y$      | $w(x) \leftarrow w(x) + w(y)$<br>$w(y) \leftarrow 0$ |
| $w(x) = w(y) > 0$ | $x > y$      | $w(x) \leftarrow w(x) + w(y)$<br>$w(y) \leftarrow 0$ |
| $w(x) = w(y) = 0$ | 모순이 발생하지 않도록 | 변화 없음                                                |

# 요약

- 연산 횟수의 하계
  - 기본 연산 정의 및 최소한의 필요한 연산을 통해 파악할 수 있음
  - 상대자(적대자) 논리를 이용하면 하계를 이해하는 데 도움이 됨
  - 상대자는 답을 찾기 어렵게 하려고 하지만, 답변에는 논리적 모순이 없어야 함
- 최댓값과 최솟값 찾기
  - 두 문제를 동시에 풀면 하나씩 푸는 것의 합보다 더 효율적이 될 수 있음
  - 최댓값은 모든 수와 비교를 거쳐야 함
  - 최솟값은 비교에서 진 것들만 대상이 됨
  - 상대자 논리: N, W, L, WL 등의 정보량,  $2(n - 2)$ 개의 정보를 가능하면 줄이는 방향으로 답함
- 두 번째로 큰 값 찾기
  - 두 번째로 큰 값은 반드시 최댓값과 비교당하였어야 함
  - 즉, 최댓값에게 패배한  $\text{ceil}(\log_2 n)$ 개의 원소들 중 두 번째로 큰 값이 있음
  - 상대자 논리: 최댓값의 가중치  $w(\max)$ 가 가능하면 천천히 증가하도록 함