Day8 하계 이론

선린인터넷고등학교 소프트웨어과
30610 나정휘
https://JusticeHui.github.io

오늘 할 일

- 강의: 하계 이론
  - 알고리즘의 최적성과 알고리즘의 하계
    - 예제) 배열의 최댓값을 찾는 문제
    - 연습 문제) 비교 기반 정렬
    - 연습 문제) 인접한 원소를 교환하는 정렬
  - 상대자 논증
    - 예제) 정렬된 배열에서 원하는 수를 찾기
  - 다른 문제로 바꿔서 하계 구하기
    - 예제) 배열에서 K번째 원소 찾기
    - 연습 문제) 볼록 껍질 구하기
- 실습: 완전 탐색 / 백트래킹 / 하계 이론 관련 문제

문제의 기본 연산

- 기본 연산
  - 주어진 문제를 해결하는데 필수적인 가장 기본적인 연산

- 알고리즘의 시간 복잡도를 분석할 때 기본 연산을 정하고, 기본 연산을 몇 번 수행하는지 횟수를 세는 경우가 많다.
  - Ex) 배열 정렬 -> 원소의 비교
  - Ex) 행렬 곱셈 -> 실수의 덧셈/곱셈

- 알고리즘의 전체 연산 횟수가 기본 연산의 수행 횟수에 비례한다면, 기본 연산의 횟수만 세는 것만으로 시간 복잡도를 구할 수 있다.

알고리즘의 최적성

- 최적성
  - 알고리즘이 더 이상 개선할 여지가 없을 만큼 최적화가 잘 되어 있는가
  - Ex) KMP 알고리즘

- 어떤 알고리즘이 최적임을 증명하는 방법
  - 알고리즘 A를 개발한다. A가 최악의 경우의 복잡도 **T(n)**을 찾는다.
  - 어떠한 알고리즘도 적어도 **F(n)**만큼의 연산을 수행해야하는 입력이 존재함을 증명한다.
  - T(n) = F(n)인지 확인한다. -> 알고리즘 A가 최악의 경우 최적이다.
    - T(n) ≠ F(n)이면 더 좋은 알고리즘이 존재하거나 더 큰 F(n)이 존재한다.

하계 이론

- 하계(Lower Bound) 이론(Theory)
  - 앞에서 말한 **F(n)**에 관심이 있는 이론
  - 하계가 무엇인가?

상계와 하계의 정의

- 원소 간의 $\leq$ 연산자가 정의된 집합 P가 있을 때

- 부분 집합 $S \subset P$의 상계 $p \in P$는 다음 성질을 만족한다.
  - 모든 $s \in S$에 대해, $s \leq p$
  - 앞에서 말한 **T(n)** - 어떠한 입력이 주어져도 이것보다 나쁘지는 않다.

- 부분 집합 $S \subset P$의 하계 $p \in P$는 다음 성질을 만족한다.
  - 모든 $s \in S$에 대해, $p \leq s$
  - 앞에서 말한 **F(n)** - 아무리 좋은 알고리즘도 이것보다 빠를 수는 없다.

알고리즘의 하계

- 길이가 n인 실수 배열의 원소 중 가장 큰 값을 구하는 문제
  - 하계를 구해보자.
  - 기본 연산: 실수 2개 비교

```
double mx = a[0]
for(int i=1; i<n; i++)
  if(a[i] > mx) mx = a[i];
print(mx)
```

- 모든 원소가 서로 다르다고 하자.
  - 모든 원소가 다르다면 n-1개의 수는 최댓값이 아니다.
  - 최댓값이 아닌 원소는 적어도 하나 이상의 다른 원소와 비교해서 최댓값이 아니라고 확정을 지어야 한다.
  - 그러므로 F(n) = n-1이다.

- 오른쪽에 있는 알고리즘도 최악의 경우 T(n) = n-1이므로 최적이다.

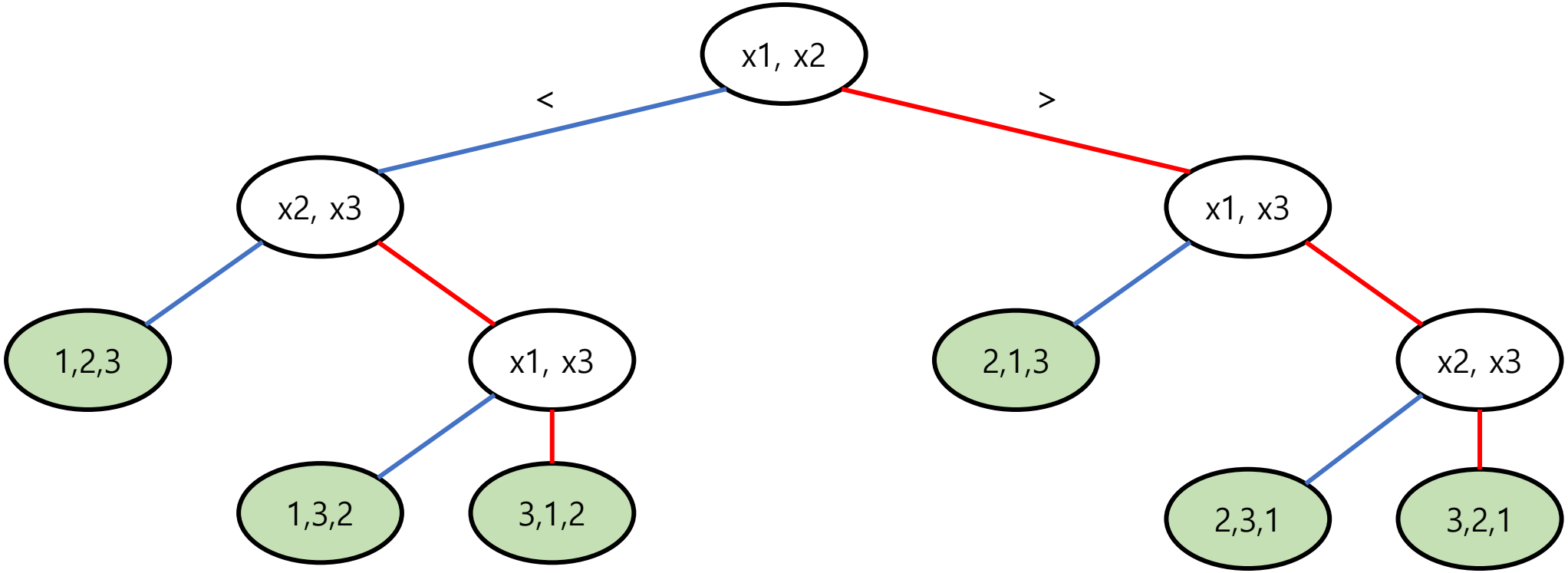
문제) 비교 기반 정렬

- 길이가 n인 실수 배열의 원소들을 오름차순으로 정렬하는 문제
  - 두 수를 비교하는 연산, 두 수의 위치를 교환하는 연산만 사용 가능
  - 기본 연산: 비교 연산
    - 교환 연산을 비교 연산보다 많이 사용하는 경우는 없다.

- Do You Know Decision Tree?

문제) 비교 기반 정렬

- 원소가 [x1, x2, x3]인 배열을 정렬하는 결정 트리를 만들자.



문제) 비교 기반 정렬

- 내부 노드는 비교, 리프 노드는 최종 결과를 나타낸다.
- 알고리즘의 수행 과정은 결정 트리의 루트 노드에서 출발해 리프 노드까지 이동하는 경로로 나타낼 수 있다.

- 어떤 입력에 대해 이 알고리즘이 비교 연산을 수행하는 횟수는 리프 노드까지 가는 경로에 있는 내부 정점의 개수이다.
  - T(n)은 리프 노드들의 깊이의 최댓값이 된다.

문제) 비교 기반 정렬

- 이진 트리의 높이를 h, 리프 노드의 개수를 m(= n!)이라고 하자.
  - m ≤ 2^h이다.
  - 양변에 log를 취해주면 $h \geq \lceil \log_2 n! \rceil \geq \log_2 n!$이다.
  - 따라서 비교 연산을 이용해 배열을 정렬할 경우, 최악의 경우에는 적어도 $\lceil \log_2 n! \rceil$번의 비교 연산을 수행해야 한다.

- $F(n) = \lceil \log_2 n! \rceil$
  - log(n!)은 Θ(n log n)이다.
  - 그러므로 비교 기반 정렬의 하계는 O(n log n)이다.

문제) 인접한 원소를 교환하는 정렬

- 길이가 n인 실수 배열의 원소들을 오름차순으로 정렬하는 문제
  - 인접한 원소 두 개를 교환하는 연산만 사용 가능
  - 기본 연산: 교환 연산

- 인접한 원소 두 개의 위치를 교환하면 해당 원소 사이의 inversion 여부가 토글된다.
  - inversion(역): i < j이면서 A[i] > A[j]인 쌍

- 길이가 n인 배열의 inversion 개수의 상한은 nC2이므로 최악의 경우에는 적어도 n(n-1)/2번의 교환 연산을 해야 한다.

상대자 논증

- 업/다운 게임?
  - A가 1부터 100까지의 자연수 중 하나를 정한다. (= x)
  - B는 A가 정한 수를 맞춰야 한다.
  - B가 어떤 수(= y)를 말하면 A는 x가 y와 같은지, 더 큰지, 더 작은지 알려준다.
  - B는 최소한의 추측으로 A가 생각한 수를 맞춰야 한다.

- 정상적인 게임 과정
  - A가 x = 26으로 정한다.
  - B가 50을 부른다. A는 정답이 50보다 작다고 말한다.
  - B가 25를 부른다. A는 정답이 25보다 크다고 말한다.
  - B가 26을 부른다. 게임을 종료한다.

상대자 논증

- A는 B와 적대적인 관계다. A는 B의 추측 횟수를 최대화 시키려고 한다. (상대자)

- 상대자 논증
  - 현재 정답이 있을 수 있는 구간은 [1, 100]이다.
  - B가 25를 부른다. 정답이 존재하는 구간을 [1, 24]와 [26, 100] 중 하나로 바꿀 수 있는데, 후자가 더 범위가 넓으니까 [26, 100]으로 바꾼다.
  - B가 90을 부른다. [26, 89]와 [91, 100] 중 하나로 바꿀 수 있는데, 전자가 더 범위가 넓으니까 [26, 89]로 바꾼다.

상대자 논증

- 상대자 논증에 대항하는 법
  - 상대자의 역할: 쿼리에 대한 결과로 “최소한의 정보”를 제공해 알고리즘을 불리하게 만듬.
  - 대처 방안: 양쪽 결과 모두 가능한 같은 양의 정보를 얻을 수 있도록 쿼리를 날린다.
    - Ex) 업/다운 게임에서는 구간의 중앙값을 부르면 상대자가 어떠한 결과를 반환해도 같은 양의 정보를 얻을 수 있다. (= 이진 탐색)

Reduction

- 주어진 문제를 다른 문제로 변환(Reduction)해서 하계를 구하는 방법도 있다.

- 서로 다른 문제 P1과 P2가 있다고 하자.
  - 만약 P1의 입력을 T(n) 시간 내에 P2의 입력으로 바꿀 수 있고
  - P2의 해로부터 T(n) 시간 내에 P1의 해를 구할 수 있으면
  - P1은 T(n) 시간에 P2로 변환된다고 한다.

- P1이 T(n) 시간에 P2로 변환되고 P1 문제의 하계가 L(n)이라면
- P2 문제의 하계는 L(n) - T(n)이다.

Reduction

- 길이가 n인 실수 배열에서 K번째로 작은 원소를 찾는 문제를 P1, 길이가 n인 실수 배열을 정렬하는 문제를 P2라고 하자.

- P1은 O(1) 시간에 P2로 변환된다.
  - 입력은 바꿀 필요가 없다.
  - P2의 해는 정렬된 배열이고, 정렬된 배열의 K번째 수는 O(1)시간에 접근할 수 있다.

문제) 볼록 껍질 구하기

- n개의 점이 주어졌을 때, Convex Hull을 구하는 알고리즘의 하계가 Ω(n log n)임을 증명하자.

- P1 = 길이가 n인 실수를 정렬하는 문제 -> Ω(n log n)
- P2 = 볼록 껍질을 구하는 문제
- P1을 O(n log n)보다 빠르게 P2로 변환할 수 있음을 보이면 된다.

문제) 볼록 껍질 구하기

- P1의 입력으로 주어진 A[1], A[2], … , A[n]가 있을 때
- (A[1], A[1]^2), … , (A[n], A[n]^2)을 P2의 입력으로 넘겨주자.

- 입력을 변환하는데 O(n)시간이 걸리고, 이 점들은 포물선 위에 위치한다. 따라서 이 점들은 모두 볼록 껍질의 꼭짓점이 된다.

- 볼록 껍질을 구한 뒤 꼭짓점들을 반시계 방향으로 순회해서 리스트를 만들어주면 볼록 껍질의 꼭짓점들을 O(n) 시간에 정렬된 배열로 만들어줄 수 있다.

문제) 볼록 껍질 구하기

- 정렬 문제의 하계가 $\Omega(n \log n)$이고, 정렬 문제를 볼록 껍질을 구하는 문제로 $O(n)$에 바꿔줄 수 있으므로 볼록 껍질을 구하는 문제의 하계는 $\Omega(n \log n - n) = \Omega(n \log n)$이다.

문제 목록