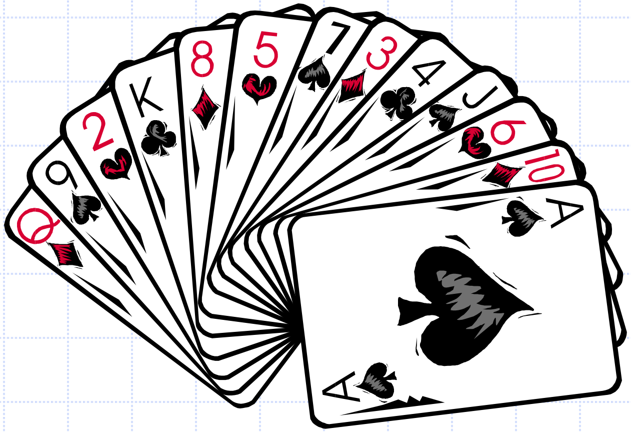
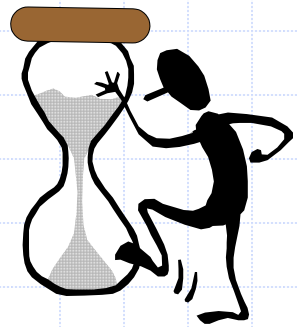
정렬 일반



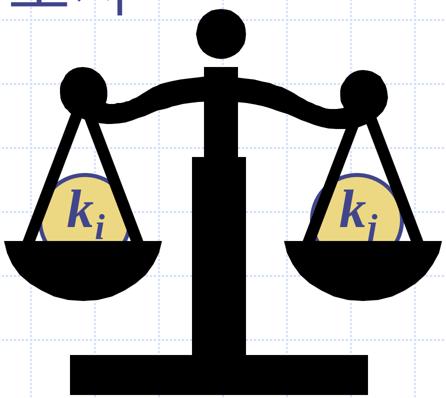
Outline

- 9.1 비교정렬의 하한
- 9.2 정렬의 안정성
- 9.3 비교정렬 알고리즘 비교
- 9.4 응용문제

비교정렬의 하한

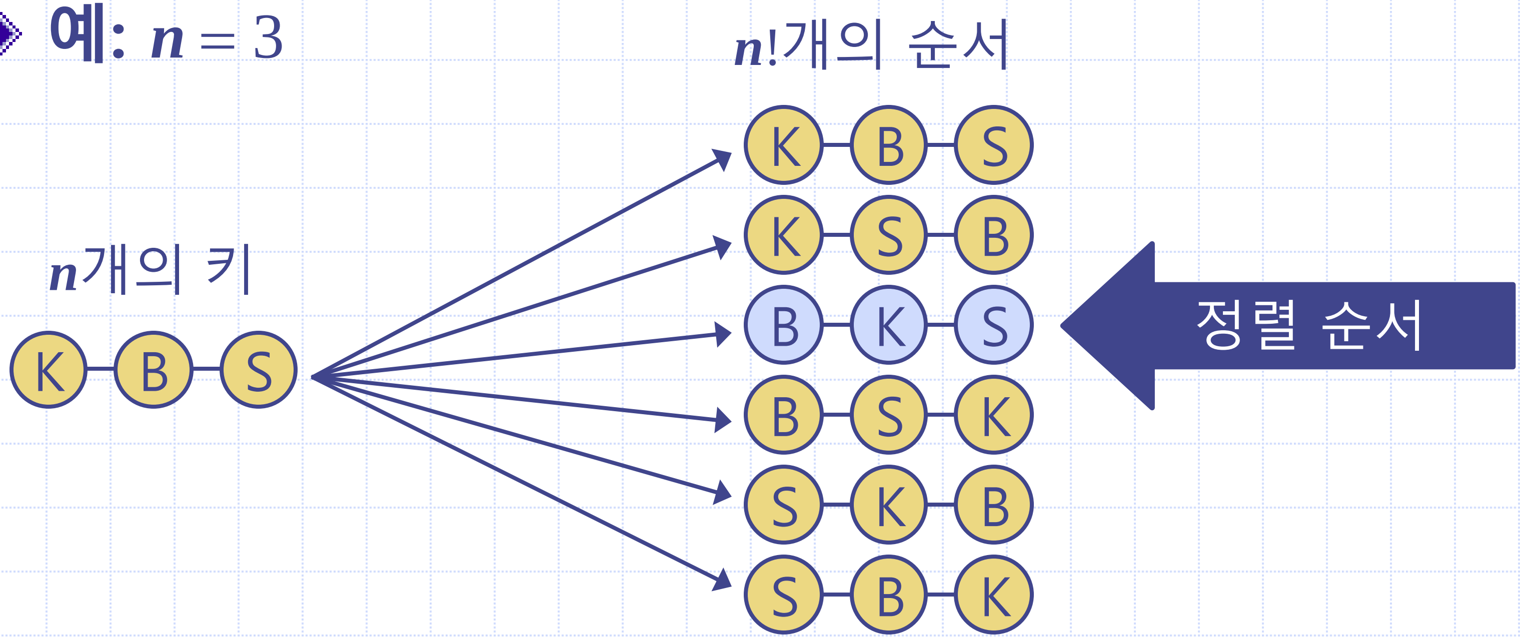


- **비교정렬**
  - **비교에 기초한 정렬**(comparison-based sorting)
  - 개체 쌍을 **비교**함으로써 정렬
  - **예:** 버블 정렬, 선택 정렬, 삽입 정렬, 힙 정렬, 합병 정렬, 퀵 정렬, …
- 각각 n개의 키 $k_1, k_2, \ldots, k_n$로 구성된 원소들을 정렬하는 **비교정렬** 알고리즘의 **하한**(lower bound)을 유도해보자



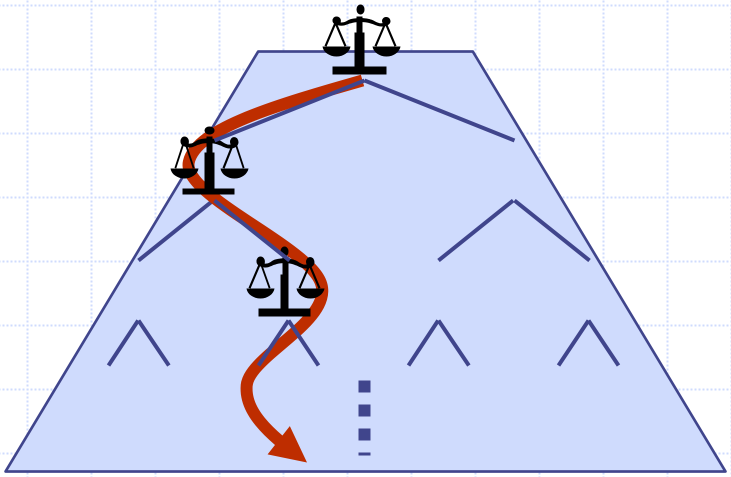
유도 단계 1

- n개의 유일한 키로부터 $n!$개의 **순서**가 존재(**참고:** 순열)
- 오름차순 정렬 기준으로, 이 가운데 **단 하나**의 순서만이 정렬 순서
- **예:** $n = 3$



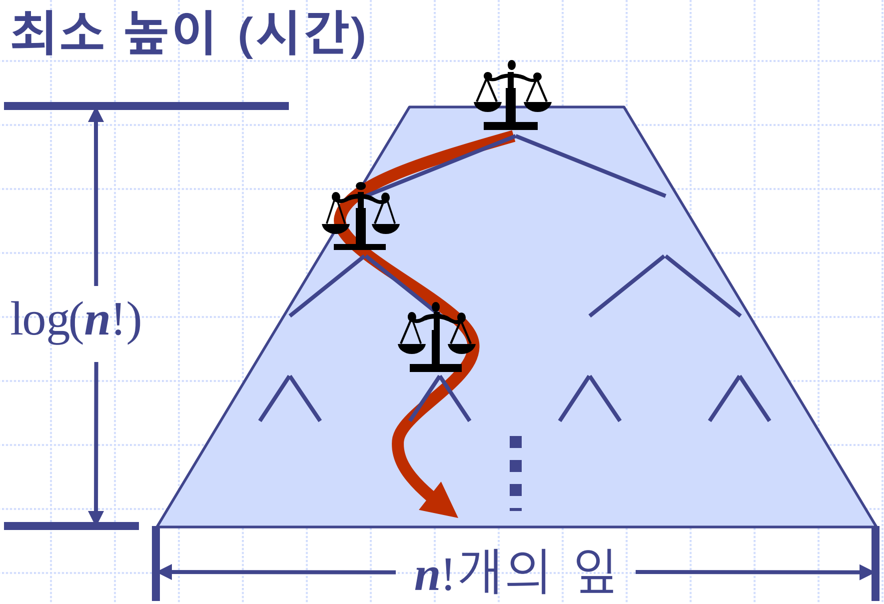
유도 단계 2

- 비교 회수를 세어보자
- 각 알고리즘이 취할 수 있는 수행은 **결정트리**(decision tree)에서 루트로부터 잎으로 향하는 경로와 일치



유도 단계 3

- 이 결정트리의 **높이:** 실행시간의 하한
- 각각의 상이한 입력 순열은 상이한 하향경로를 순회
- ***n***!개의 외부노드가 있으므로, 높이는 최소 log(***n***!)



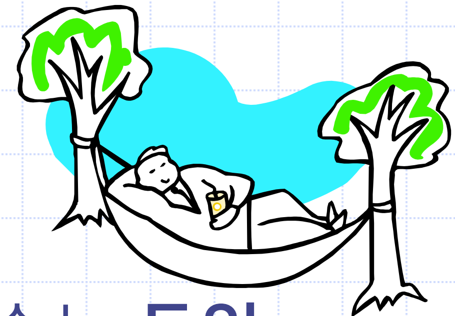
유도 단계 4

- 그러므로 어떤 **비교정렬** 알고리즘도 최소 $\log(n!)$ 시간을 소요
- 그리고,*

$$\log(n!) \geq \log(n/2)^{n/2} = (n/2)\log(n/2)$$

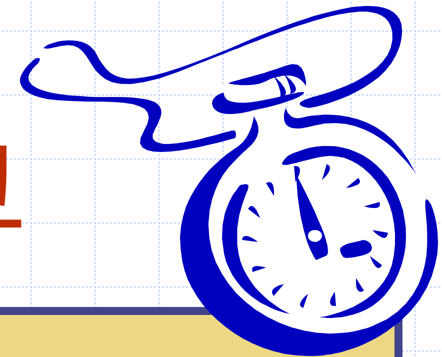
- **결론:** 어떤 **비교정렬** 알고리즘이라도 $\Omega(n \log n)$ 시간에 수행

정렬의 안정성



- 키-원소 항목들을 정렬할 때, 중요한 이슈는 **동일 키**가 어떻게 처리되는냐는 것이다
- $L = ((k_0, e_0), \ldots, (k_{n-1}, e_{n-1}))$을 항목들의 리스트라 하자
- 두 개의 항목 (k_i, e_i)과 (k_j, e_j)에 대해,
  - $k_i = k_j$며 정렬 전에 (k_i, e_i)가 (k_j, e_j) 보다 앞서 있었다면(즉, $i < j$),
  - 만약 정렬 후에도 (k_i, e_i)가 (k_j, e_j)보다 앞서 있다면,
  - 그 정렬 알고리즘을 **안정적**(stable)이라고 말한다
- 정렬 알고리즘에 있어서 **안정성**(stability)은 중요 – 왜냐면 많은 응용에서 동일 키 원소들의 원래 순서가 보존되어야 할 필요가 있기 때문

비교정렬 알고리즘 비교



	시간	주요 전략	비고
선택 정렬	**O**(n^2)	◈ 우선순위 큐 (무순 리스트로 구현)	◈ 제자리 ◈ 느림 (소규모 입력에 적당)
삽입 정렬	**O**(n^2)	◈ 우선순위 큐 (순서 리스트로 구현)	◈ 제자리 ◈ 느림 (소규모 입력에 적당)
힙 정렬	**O**($n \log n$)	◈ 우선순위 큐 (힙으로 구현)	◈ 제자리 ◈ 빠름 (대규모 입력에 적당)
합병 정렬	**O**($n \log n$)	◈ 분할통치	◈ 순차 데이터접근 ◈ 빠름 (초대규모 입력에 적당)
퀵 정렬	**O**($n \log n$) 기대시간	◈ 분할통치	◈ 제자리, 무작위 ◈ 가장 빠름 (대규모 입력에 적당)

응용문제: 투표

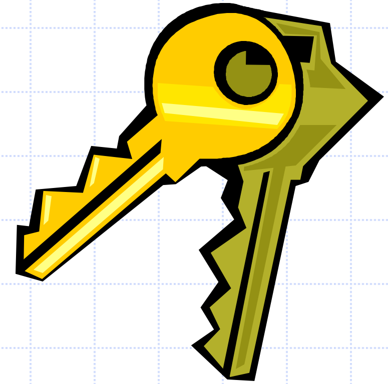
- ***n***-원소 리스트 ***L***이 주어졌다고 가정하자 – 여기서 ***L***의 각 원소는 선거에서의 **투표**를 표현
- 각 투표는 선택된 후보자의 기호를 나타내는 **정수**로 주어진다
  - 기호들은 정수지만 빠진 번호가 있을 수도 있다
- 어떤 번호가 빠졌는지 또는 후보자가 모두 몇 명이나 되는지에 대한 아무런 정보 없이, ***L***이 나타내는 투표 내용에서 당선자를 찾아내는 **O(*n*** log ***n*)**-시간 메쏘드를 작성하라
- **전제:** 가장 많은 표를 획득한 후보자가 당선된다
- **예:** 아래 투표 리스트에서 기호 7이 당선자다

4 — 7 — 1 — 7 — 3 — 12 — 1 — 7

해결

- 메쏘드
  1. 먼저 리스트 L을 후보자의 기호 순서로 정렬
  2. 정렬된 리스트를 순회하면서, 현재까지 최대 득표한 기호와 득표수를 저장
  3. 각 기호에 대해, 현재까지의 최대 득표수와 비교하여 필요하다면 이를 갱신
- 실행시간
  - 1단계: $\mathbf{O}(n \log n)$
  - 2단계: $\mathbf{O}(n)$
  - 그러므로 총 $\mathbf{O}(n \log n)$

응용문제: 두 키로 정렬



- n개의 (학생 이름, 점수) 쌍으로 구성된 **무순**의 리스트가 있다 – 여기서 **점수**는 0에서 100 사이의 정수며 **학생 이름**은 문자열로 표현되어 있다
- 이 데이터를 **점수**의 내림차순으로 정렬하되, 점수가 같은 경우 **학생 이름**의 오름차순으로 정렬하고자 한다(오른쪽 표 참고)
- 이와 같이 정렬하기 위해 각 정렬 키에 대해 어떤 순서로, 어떤 **비교정렬** 알고리즘을 사용할지 설명하라
- **전제:** 전체 정렬 작업은 최악의 경우 $O(n \log n)$ 시간에 수행되어야 한다

심청	95
콩쥐	90
장화	86
홍련	86
연흥부	74
배비장	65
연놀부	65
팥쥐	65
변학도	41

해결

- 다음 1, 2 단계로 나누어 진행

1. **학생 이름**을 정렬 키로 사용하여 오름차순 **힙 정렬**(또는 **합병 정렬**)을 수행 – 이 단계에서 정렬의 안전성은 중요하지 않다
2. **정렬의 안정성**을 고려하여 알고리즘을 선택 – 1단계의 정렬 결과에, **점수**를 키로 사용하여 내림차순 **합병 정렬**을 수행