자료구조

L09: Sorting (4)

2022년 1학기

국민대학교 소프트웨어학부

Overview

- Empirical Comparison
- Sorting Lower Bound

Empirical Comparison

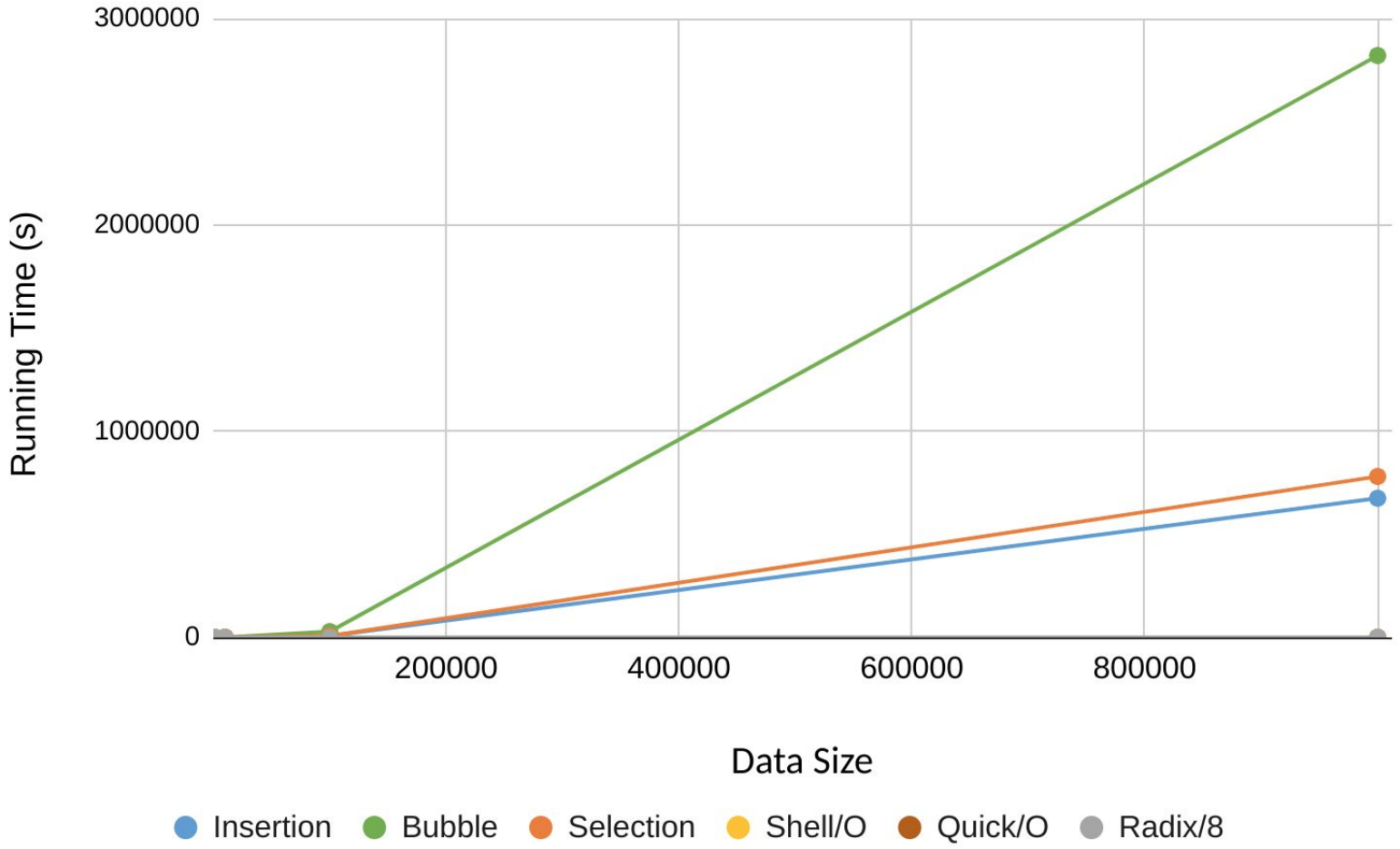
Sort	10	100	1K	10K	100K	1M	Up	Down
							10K	
Insertion	0.00023	0.007	0.66	64.98	7281.00	674420	0.04	129.05
Bubble	0.00035	0.020	2.25	277.94	27691.00	2820680	70.64	108.69
Selection	0.00039	0.012	0.69	72.47	7356.00	780000	69.76	69.58
Shell	0.00034	0.008	0.14	1.99	30.20	554	0.44	0.79
Shell/O	0.00034	0.008	0.12	1.91	29.00	530	0.36	0.64
Merge	0.00050	0.010	0.12	1.61	19.30	219	0.83	0.79
Merge/O	0.00024	0.007	0.10	1.31	17.20	197	0.47	0.66
Quick	0.00048	0.008	0.11	1.37	15.70	162	0.37	0.40
Quick/O	0.00031	0.006	0.09	1.14	13.60	143	0.32	0.36
Heap	0.00050	0.011	0.16	2.08	26.70	391	1.57	1.56
Heap/O	0.00033	0.007	0.11	1.61	20.80	334	1.01	1.04
Radix/4	0.00838	0.081	0.79	7.99	79.90	808	7.97	7.97
Radix/8	0.00799	0.044	0.40	3.99	40.00	404	4.00	3.99

Empirical Comparison

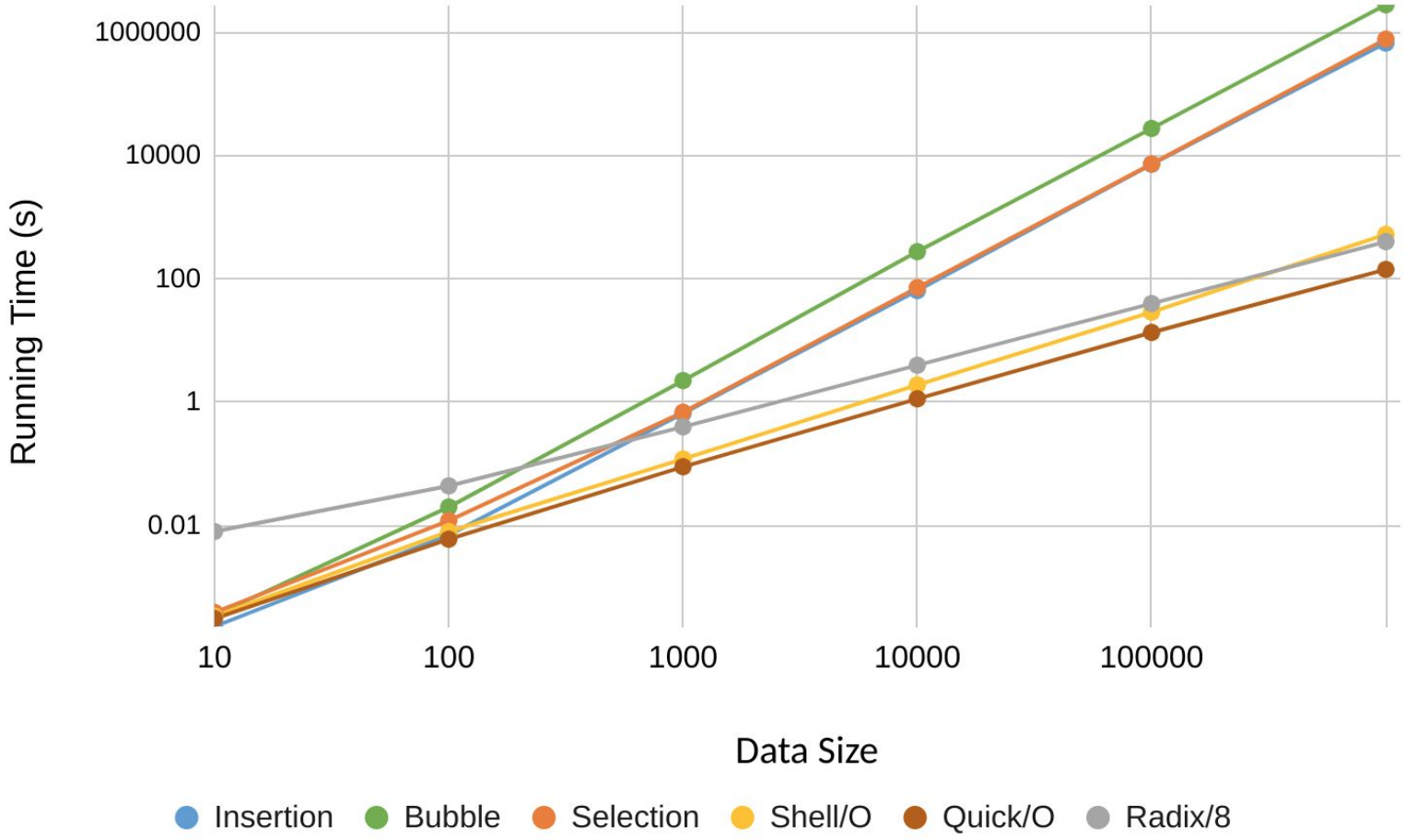
10K

Sort	10	100	1K	10K	100K	1M	Up	Down
Insertion	0.00023	0.007	0.66	64.98	7281.00	674420	0.04	129.05
Bubble	0.00035	0.020	2.25	277.94	27691.00	2820680	70.64	108.69
Selection	0.00039	0.012	0.69	72.47	7356.00	780000	69.76	69.58
Shell	0.00034	0.008	0.14	1.99	30.20	554	0.44	0.79
Shell/O	0.00034	0.008	0.12	1.91	29.00	530	0.36	0.64
Merge	0.00050	0.010	0.12	1.61	19.30	219	0.83	0.79
Merge/O	0.00024	0.007	0.10	1.31	17.20	197	0.47	0.66
Quick	0.00048	0.008	0.11	1.37	15.70	162	0.37	0.40
Quick/O	0.00031	0.006	0.09	1.14	13.60	143	0.32	0.36
Heap	0.00050	0.011	0.16	2.08	26.70	391	1.57	1.56
Heap/O	0.00033	0.007	0.11	1.61	20.80	334	1.01	1.04
Radix/4	0.00838	0.081	0.79	7.99	79.90	808	7.97	7.97
Radix/8	0.00799	0.044	0.40	3.99	40.00	404	4.00	3.99

Empirical Comparison



Empirical Comparison



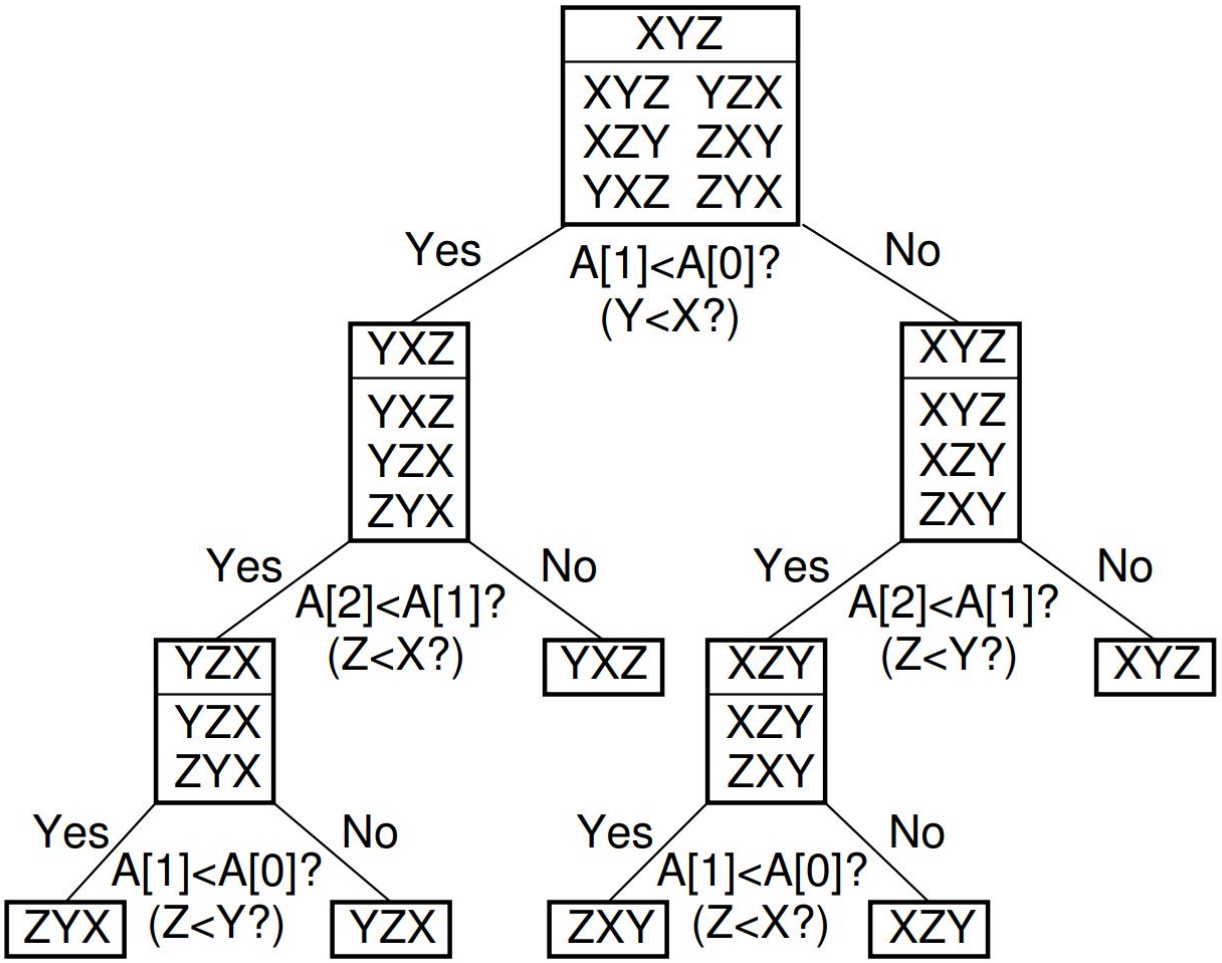
Empirical Comparison: Observation

- Growth rate: O(n)과 $O(n^2)$의 차이
- $O(n^2)$ 알고리즘들은 n이 클 때 매우 느리다.
  - n이 작을 땐 잘 동작한다.
- Insertion sort는 'Up' 일 때 아주 빠르다.
- 일반적으로, quicksort와 merge sort가 가장 빠른 성능을 보인다.
- Radix sort
  - O(nlogn) 알고리즘보다 그다지 빠르지 않다.
  - Radix/8이 Radix/4보다 공간은 조금 더 쓰지만 빠르다.

Sorting Lower Bound

- 정렬 문제의 upper bound = $O(n\log n)$
  - Why?
- 정렬 문제의 lower bound = $\Omega(n)$
  - 데이터 전체를 읽는데 걸리는 시간은 필요하므로.
- 더 빠른 정렬 알고리즘이 존재할까?
  - 두 가지 방법
    - 1. 복잡도가 $O(n\log n)$보다 작은 알고리즘을 찾는다
    - 2. 정렬 문제의 lower bound가 $\Omega(n\log n)$ 임을 증명한다

Decision Trees



Lower Bound Proof

- n!개의 순열이 존재
- 정렬 알고리즘은 n!의 순열 중에 완전히 정렬된 하나의 순열을 찾아가는 과정으로 생각할 수 있음
- Decision Tree의 각 leaf node는 하나의 순열에 해당됨
- 모든 비교기반 정렬 알고리즘은 n!개의 leaf node를 가짐

n개의 leaf node를 갖는 binary tree의 최소 높이(height)는?

Lower Bound Proof

- n개의 leaf node를 갖는 binary tree 높이의 lower bound: $\Omega(\log n)$
- n!개의 leaf node를 갖는 binary tree 높이의 lower bound: $\Omega(\log n!) = \Omega(n \log n)$
  - Stirling's approximation: $n! \approx \sqrt{2\pi n} \times \left(\frac{n}{e}\right)^n$
  - logn! = log1 + log2 + … + logn <= nlogn
- decision tree의 어떤 노드가 worst case에 해당할까?
  - 가장 낮은 level에 속한 node
  - 즉, 모든 알고리즘의 최악의 경우 복잡도: $\Omega(n \log n)$

What You Need to Know

- Sorting: puts elements in a certain order
  - Evaluation: # swaps, # comparisons
- Sorting algorithms
  - Best: $O(n \log n)$ algorithms (e.g., quicksort, merge sort)
  - Some algorithms better than the best ones for special cases
    - Heapsort, Binsort, radix sort
  - Cost analysis
- Lower bound analysis

Questions?