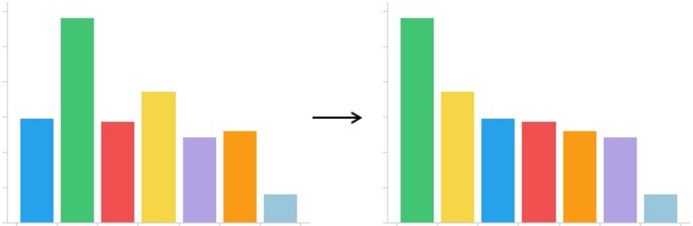
Data Structures:
Internal Sorting: Insertion Sort, Selection Sort, Merge Sort, Quick Sort

Won Kim
(Lecture by Youngmin Oh)
Spring 2022

Sorting - Internal

Sorting

- Definition
  - Arranging elements of a list in some order
    - Lexicographical order (symbols, numbers)
    - Ascending order or descending order
    - Composite-key sort
- Why sort?
  - Required end result
    - Easy visual search
  - Efficient intermediate data structure for computing the end result
- In general, entire records are sorted and change positions.



Motivating Example 1

name	age	salary	employer
Kim	25	200	LG
Lee	30	300	IBM
Park	23	250	LG
Cho	40	500	Samsung
…			
Chung	28	900	Samsung

Motivating Example 1 (cont'd)

name	age	salary	employer
Cho	40	500	Samsung
Chung	28	900	Samsung
Kim	25	200	LG
Lee	30	300	IBM
...			
Park	23	250	LG

This is easier to search by name

Motivating Example 2

name	age	salary	employer
Kim	25	200	LG
Lee	30	300	IBM
Park	23	250	LG
Cho	40	500	Samsung
…			
Chung	28	900	Samsung

Who works for Samsung?

Who is the
- oldest?
- highest paid?

Motivating Example 2 (cont'd)

name	age	salary	employer
Chung	28	900	Samsung
Cho	40	500	Samsung
Park	23	250	LG
Kim	25	200	LG
…			
Lee	30	300	IBM

Who works for Samsung?

Who is the
- oldest?
- highest paid?

What the Sorted Results Make Possible

- Speedy sequential search (that can stop in the middle of a search)
- Binary search ($O(\log_2 n)$ performance)
- Fast join of multiple tables (in a relational database)

Join: What is the number of employees of the company where Cho works?

To find the answer, we need to join two tables: employee and company.

employee

name	age	salary	employer
Kim	25	200	LG
Lee	30	300	IBM
Park	23	250	LG
Cho	40	500	Samsung
...			
Chung	28	900	Samsung

company

company	years	country	employees
IBM	107	USA	450,000
LG	60	Korea	38,000
Samsung	49	250	310,000
SAP	46	500	15,000

Sorting Algorithms

- Internal Sorting
  - List to be sorted fits in main memory.
- External Sorting
  - List to be sorted resides on secondary storage.

Sorting Algorithms

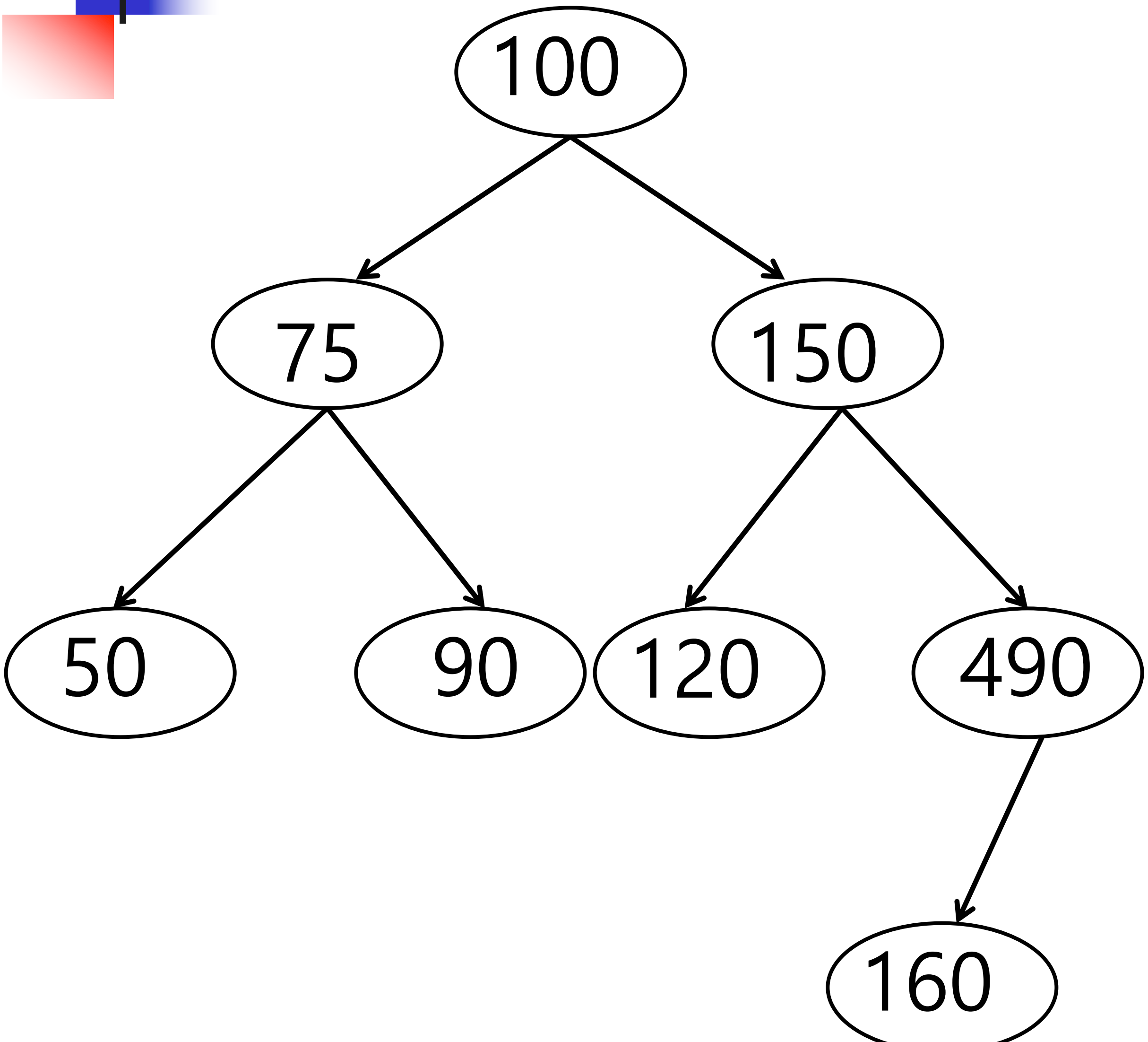
- Internal Sorting
  - Insertion sort
  - Selection sort
  - (Bubble sort – worst; don't even think about it)

  - Merge sort
  - Quick sort

  - Heap sort

  - Radix sort

Inorder Traversal of a Binary Search Tree: Outputs Keys in Ascending Sort Order



50
75
90
100
120
150
160
490

Insertion Sort

Insertion Sort

- Start with a sorted list.
- To insert a new key, search the list for a correct position, and insert it there.

Insertion Sort: Example

insert: 170 (170)
insert: 90 (90, 170)
insert: 2 (2, 90, 170)
insert: 802 (2, 90, 170, 802)
insert: 24 (2, 24, 90, 170, 802)
insert: 45 (2, 24, 45, 90, 170, 802)
insert: 75 (2, 24, 45, 75, 90, 170, 802)
insert: 66 (2, 24, 45, 66, 75, 90, 170, 802)

Insertion Sort: Exercise

sort the list:

22, 25, 7, 3, 30, 11, 14, 8

Insertion Sort: Exercise

sort the list:

22, 25, 7, 3, 30, 11, 14, 8

insert: 22 (22)
insert: 25 (22, 25)
insert: 7 (7, 22, 25)
insert: 3 (3, 7, 22, 25)
insert: 30 (3, 7, 22, 25, 30)
insert: 11 (3, 7, 11, 22, 25, 30)
insert: 14 (3, 7, 11, 14, 22, 25, 30)
insert: 8 (3, 7, 8, 11, 14, 22, 25, 30)

Insertion Sort: Performance and Qualities

- $O(n^2)$
  - n = the number of new keys to be inserted into an empty or existing sorted list
  - time to search for the correct position for a new key on the current sorted list
  - time to move the bigger keys on the current sorted list to make room for a new key
- Simple and good for a short list
- Good when the list is already partially sorted.

Selection Sort

Selection Sort

- n-1 passes over a list of n keys
- On the i^{th} pass
  - select the largest key among the first n-i keys, and exchange it with the $n-i+1^{th}$ key
- Reading
  - http://en.wikipedia.org/wiki/Selection_sort

Selection Sort: Example

sort the list:

13, 4, 9, 21, 37, 17, 22, 3, 8

pass 1: 13, 4, 9, 21, 8, 17, 22, 3, 37

pass 2: 13, 4, 9, 21, 8, 17, 3, 22, 37

pass 3: 13, 4, 9, 3, 8, 17, 21, 22, 37

pass 8: 3, 4, 8, 9, 13, 17, 21, 22, 37

Selection Sort: Exercise

sort the list:

22, 25, 7, 3, 30, 11, 14, 8

Pass 1: 22, 25, 7, 3, 8, 11, 14, 30

Pass 2: 22, 14, 7, 3, 8, 11, 25, 30

Pass 3: 11, 14, 7, 3, 8, 22, 25, 30

Pass 4: 11, 8, 7, 3, 14, 22, 25, 30

Pass 5: 3, 8, 7, 11, 14, 22, 25, 30

Pass 6: 3, 7, 8, 11, 14, 22, 25, 30

Pass 7: 3, 7, 8, 11, 14, 22, 25, 30

Duplex Selection Sort

- (n-1)/2 passes over a list of n keys
  - Select the largest key and the smallest key, and
  - Exchange the largest key with the last key, and
  - Exchange the smallest key with the first key.
- $O(n^2)$ performance
  - n is the number of keys in the list

Selection Sort: Performance

- $O(n^2)$ performance
  - n is the number of keys in the list
  - n^2 comparisons
  - generally worse than insertion sort
  - O(n) exchanges
- Useful for a small list stored in EEPROM or Flash.
  - "exchanges" require writing to an array.
  - For an array stored in EEPROM or Flash, writing to memory is significantly more expensive than reading.

Merge Sort

Merge Sort

- Divide and Conquer
- Split and Merge
  - Split: Divide a list successively into two sorted (X) sub-lists.
  - Merge: Merge the two sub-lists successively into a single sorted list.
- Invented by John von Neumann in 1945
- Supplementary Reading
  - http://en.wikipedia.org/wiki/Merge_sort

Merge Example

merge two sorted lists:

15, 20, 25, 30, 35 7, 18, 22, 28, 34

↑ ↑

7

15, 20, 25, 30, 35 7, 18, 22, 28, 34

↑ ↑

7, 15

15, 20, 25, 30, 35 7, 18, 22, 28, 34

↑ ↑

7, 15, 18

Merge Example (cont'd)

15, 20, 25, 30, 35 7, 18, 22, 28, 34

7, 15, 18, 20

15, 20, 25, 30, 35 7, 18, 22, 28, 34

7, 15, 18, 20, 22

15, 20, 25, 30, 35 7, 18, 22, 28, 34

7, 15, 18, 20, 22, 25

Merge Example (cont'd)

15, 20, 25, 30, 35　　　　7, 18, 22, 28, 34

7, 15, 18, 20, 22, 25, 28

15, 20, 25, 30, 35　　　　7, 18, 22, 28, 34

7, 15, 18, 20, 22, 25, 28, 30

15, 20, 25, 30, 35　　　　7, 18, 22, 28, 34

7, 15, 18, 20, 22, 25, 28, 30, 34, 35

Exercise: Merge

merge two sorted lists:

3, 9, 20, 25, 35, 40

7, 18, 22, 28, 34

Exercise: Merge

merge two sorted lists:

3, 9, 20, 25, 35, 40 7, 18, 22, 28, 34

3

3, 9, 20, 25, 35, 40 7, 18, 22, 28, 34

3, 7

3, 9, 20, 25, 35, 40 7, 18, 22, 28, 34

3, 7, 9

Exercise: Merge

merge two sorted lists:

3, 9, 20, 25, 35, 40 7, 18, 22, 28, 34

3, 7, 9, 18

3, 9, 20, 25, 35, 40 7, 18, 22, 28, 34

3, 7, 9, 18, 20

3, 9, 20, 25, 35, 40 7, 18, 22, 28, 34

3, 7, 9, 18, 20, 22

Exercise: Merge

merge two sorted lists:

3, 9, 20, 25, 35, 40 7, 18, 22, 28, 34

3, 7, 9, 18, 20, 22, 25

3, 9, 20, 25, 35, 40 7, 18, 22, 28, 34

3, 7, 9, 18, 20, 22, 25, 28

3, 9, 20, 25, 35, 40 7, 18, 22, 28, 34

3, 7, 9, 18, 20, 22, 25, 28, 34

Exercise: Merge

merge two sorted lists:

3, 9, 20, 25, 35, 40 7, 18, 22, 28, 34

↑

3, 7, 9, 18, 20, 22, 25, 28, 34, 35

3, 9, 20, 25, 35, 40 7, 18, 22, 28, 34

↑

3, 7, 9, 18, 20, 22, 25, 28, 34, 35, 40

Merge Sort

sort the list:

22, 25, 7, 3, 30, 11, 14, 8

splitting phase

22, 25, 7, 3

30, 11, 14, 8

22, 25

7, 3

30, 11

14, 8

22

25

7

3

30

11

14

8

Merge Sort: (cont'd)

merging phase

22 | 25 | 7 | 3 | 30 | 11 | 14 | 8

pass 1:

22, 25 | 3, 7 | 11, 30 | 8, 14

pass 2:

3, 7, 22, 25 | 8, 11, 14, 30

pass 3:

3, 7, 8, 11, 14, 22, 25, 30

Merge Sort: Merging an Odd Number of Data

merging phase

22	25	7	3	30	11	14	8	13

22, 25	3, 7	11, 30	8, 14

3, 7, 22, 25	8, 11, 14, 30

3, 7, 8, 11, 14, 22, 25, 30

3, 7, 8, 11, 13, 14, 22, 25, 30

Merge Sort: Performance and Qualities

- $O(n \log_2 n)$ comparisons and copying of keys
  - n: number of keys in the list
  - $\log_2 n$ times (guaranteed) – average and worst case
  - best "worst case" sorting
- $O(n)$ storage space
- Stable Sort
- Amenable to parallel processing
- Amenable to adaptation for external sorting

Performance Comparison

N^2 vs. $N \log_2 N$

N^2

	$n = 10^3$	$n = 10^6$	$n = 10^9$
PC		2.8 hr	317 yr
super com		1 sec	1.7 wk

$N \log_2 N$

	$n = 10^3$	$n = 10^6$	$n = 10^9$
PC		1 sec	18 min
super com			

pc: 10^8 compares/sec
supercom: 10^{12} compares/sec

Quick Sort

Quick Sort

- One of the most widely used sorting algorithms
- Sort by divide and conquer
- Invented by C.A.R. Hoare in 1961
- Types of quick sort
  - Plain quick sort (creates new lists for intermediate results)
  - In-place quick sort (uses the original list)
- Supplementary Reading
  - http://en.wikipedia.org/wiki/Quicksort
  - http://www.cs.purdue.edu/homes/ayg/CS251/slides/chap8b.pdf

Quick Sort

- Select a Pivot.
  - Arbitrary item on the list (first, last, middle,…)
- Partition
  - Move Keys < Pivot into a L-List.
  - Move Keys = Pivot into a P-List.
  - Move Keys > Pivot into a R-List.
- Recurse on the L-List, and R-List, until both lists are sorted.
- Concatenate L-List, P-List, R-List into a new list.

Quick Sort Pseudo Code

```
QUICKSORT(A, p, r)
if p < r
    q = PARTITION(A, p, r)
    QUICKSORT(A, p, q - 1)
    QUICKSORT(A, q + 1, r)
```

```
PARTITION(A, p, r)
x = A[r]
i = p - 1
for j = p to r - 1
    if A[j] ≤ x
        i = i + 1
        exchange A[i] with A[j]
exchange A[i + 1] with A[r]
return i + 1
```

Quick Sort: Example 1

sort the list: (pivot=first element)

70, 90, 12, 80, 24, 45, 75, 66

pass 1: 12, 24, 45, 66, 70, 90, 80, 75

pass 2: 12, 24, 45, 66, 70, 80, 75, 90

pass 3: 12, 24, 45, 66, 70, 75, 80, 90

pass 4: 12, 24, 45, 66, 70, 75, 80, 90

Quick Sort: Exercise

sort the list:(pivot=first element)

17, 90, 2, 80, 14, 13, 75, 16

Quick Sort: Solution

sort the list:

17, 90, 2, 80, 14, 13, 75, 16

pass 1: 2, 14, 13, 16, 17, 90, 80, 75

pass 2: 2, 14, 13, 16, 17, 80, 75, 90

pass 3: 2, 13, 14, 16, 17, 75, 80, 90

Quick Sort: Exercise

sort the list:(pivot=last element)

17, 90, 2, 80, 14, 13, 75, 16

Quick Sort: Exercise

sort the list:(pivot=last element)
17, 90, 2, 80, 14, 13, 75, 16

pass 1: 2, 14, 13, 16, 17, 90, 80, 75

pass 2: 2, 13, 14, 16, 17, 75, 90, 80

pass 3: 2, 13, 14, 16, 17, 75, 80, 90

Quick Sort: Example 2 (worst case)

sort the list: (pivot=first element)

2, 24, 45, 66, 75, 90, 170, 802

pass 1: 2, 24, 45, 66, 75, 90, 170, 802
pass 2: 2, 24, 45, 66, 75, 90, 170, 802
pass 3: 2, 24, 45, 66, 75, 90, 170, 802
pass 4: 2, 24, 45, 66, 75, 90, 170, 802
pass 5: 2, 24, 45, 66, 75, 90, 170, 802
pass 6: 2, 24, 45, 66, 75, 90, 170, 802
pass 7: 2, 24, 45, 66, 75, 90, 170, 802

In-Place Quick Sort

- Use the last element in the list as pivot for ascending order sorting.
- In the Partition Step
  - Use 2 cursors (L cursor and R cursor)
    - L cursor scans the list from left to right
    - R cursor scans the list from right to left
  - swap left key > pivot with right key < pivot
  - When the L cursor and R cursor cross, only move only one of them.
    - Move the L cursor until it finds a key > pivot or reaches the pivot
    - Swap the last key found with the pivot, if necessary

In-Place Quick Sort: Concepts

sort the list: (pivot=last element)

85, 24, 63, 45, 17, 31, 96, 50

L ⟶ ⟵ R

look for key>pivot look for key<pivot

85, 24, 63, 45, 17, 31, 96, 50

swap two keys so that

key<pivot will go to the left and

key>pivot will go to the right

In-Place Quick Sort Example

sort the list:

85, 24, 63, 45, 17, 31, 96, 50

L ⟶ ⟵ R

pass 1: 85, 24, 63, 45, 17, 31, 96, 50

31, 24, 63, 45, 17, 85, 96, 50

exchange 31 and 85

and continue search

L ⟶ ⟵ R

In-Place Quick Sort Example (cont'd)

sort the list:

85, 24, 63, 45, 17, 31, 96, 50

L ⟶ ⟵ R

pass 1: 85, 24, 63, 45, 17, 31, 96, 50

31, 24, 63, 45, 17, 85, 96, 50

31, 24, 63, 45, 17, 85, 96, 50

31, 24, 17, 45, 63, 85, 96, 50

L ⟶

⟵ R

R stops, but L continues until it finds a key > pivot

The last key found by L is 63.

It is > pivot (50), so swap it with the pivot.

Exercise: In-Place Quick Sort

- Continue and complete the example

  sort the list: (pivot=last element)

  85, 24, 63, 45, 17, 31, 96, 50

pass 1 result	31, 24, 17, 45, 50, 85, 96, 63
pass 2:	31, 24, 17, 45, 50, 63, 96, 85
pass 3:	17, 24, 31, 45, 50, 63, 85, 96
pass 4:	17, 24, 31, 45, 50, 63, 85, 96

Exercise (show all the steps)

sort the list: (pivot=last element)
(* do not split up the P list already formed *)

85, 17, 63, 45, 17, 31, 85, 50

Exercise (show all the steps)

sort the list: (pivot=last element)

(* do not split up the P list already formed *)

50 63

85, 17, 63, 45, ~~17~~, 31, 85, ~~50~~

31 17 63 85

Pass 1: 31, 17, 17, 45, 50, 85, 85, 63

Pass 2: 31, 17, 17, 45, 50, 63, 85, 85

Pass 3: 17, 17, 31, 45, 50, 63, 85, 85

Pass 4: 17, 17, 31, 45, 50, 63, 85, 85

Exercise (show all the steps)

- The first element of the list may be selected as pivot to do descending order sorting.
- In this case, the description on page 45 (the L cursor and R cursor) is reversed in direction.
- ** This is left as exercise.

sort the list: (pivot=first element)

85, 24, 63, 45, 17, 31, 96, 50

Quick Sort: Pivot Selection

- First or Last
- Median of Three
  - Select 3 keys, and select the middle (sized) key.
    - first 3, last 3, random 3
    - (first, last, middle)
    - median of median of three

Quick Sort: Performance and Problems

- $O(n \log_2 n)$
  - n comparisons & copying of keys on each pass
  - about $\log_2 n$ passes
    - if the distribution of the keys is reasonably balanced around the pivot
- $O(n)$ extra storage space for the temporary lists
- Terrible on an already sorted or nearly sorted list
  - worst case: $O(n^2)$
  - Use quick sort with randomized pivot to avoid the worst case performance

Quick Sort vs. Merge Sort

	quick sort	merge sort
worst case	$O(n^2)$	$O(n \log_2 n)$
best case	$O(n \log_2 n)$	$O(n \log_2 n)$
average	$O(n \log_2 n)$	$O(n\log_2 n)$
storage	$O(n)$	$O(n)$
stable?	no	yes

End of Lecture