Computer Science 112
Data Structures

Lecture 25:
Review for Final Exam

Merge (*part* of merge sort)

- **Merging:**
  - **Start with two lists, each in order within itself**
  - **Produce one combined, in-order list**

2 5 8 9

1 3 4 6

=> 1 2 3 4 5 6 8 9

Merging

- **Merging:**

  Inputs:

  $\boxed{2}$ **5 8 9**

  $\boxed{1}$ **3 4 6**

  Output:

Merging

- **Merging:**

  Inputs:

  $\boxed{2}$ 5 8 9

  1 $\boxed{3}$ 4 6

  Output:

  1

Merging

- **Merging:**

  Inputs:

  2 $\boxed{5}$ 8 9

  1 $\boxed{3}$ 4 6

  Output:

  1 2

Merging

- **Merging:**

  Inputs:

  2 $\boxed{5}$ 8 9

  1 3 $\boxed{4}$ 6

  Output:

  1 2 3

Merging

- **Merging:**

  Inputs:

  2 $\boxed{5}$ 8 9

  1 3 4 $\boxed{6}$

  Output:

  1 2 3 4

Merging

- **Merging:**

  Inputs:

  2 5 $\boxed{8}$ 9

  1 3 4 $\boxed{6}$

  Output:

  1 2 3 4 5

Merging

- **Merging:**

  Inputs:

  2 5 $\boxed{8}$ 9

  1 3 4 6 $\boxed{}$

  Output:

  1 2 3 4 5 6

Merging

Merging:

Inputs:

2 5 8 9

1 3 4 6

Output:

1 2 3 4 5 6 8 9

How much work?

Merging

- **Work: worst case O(Length) compares**

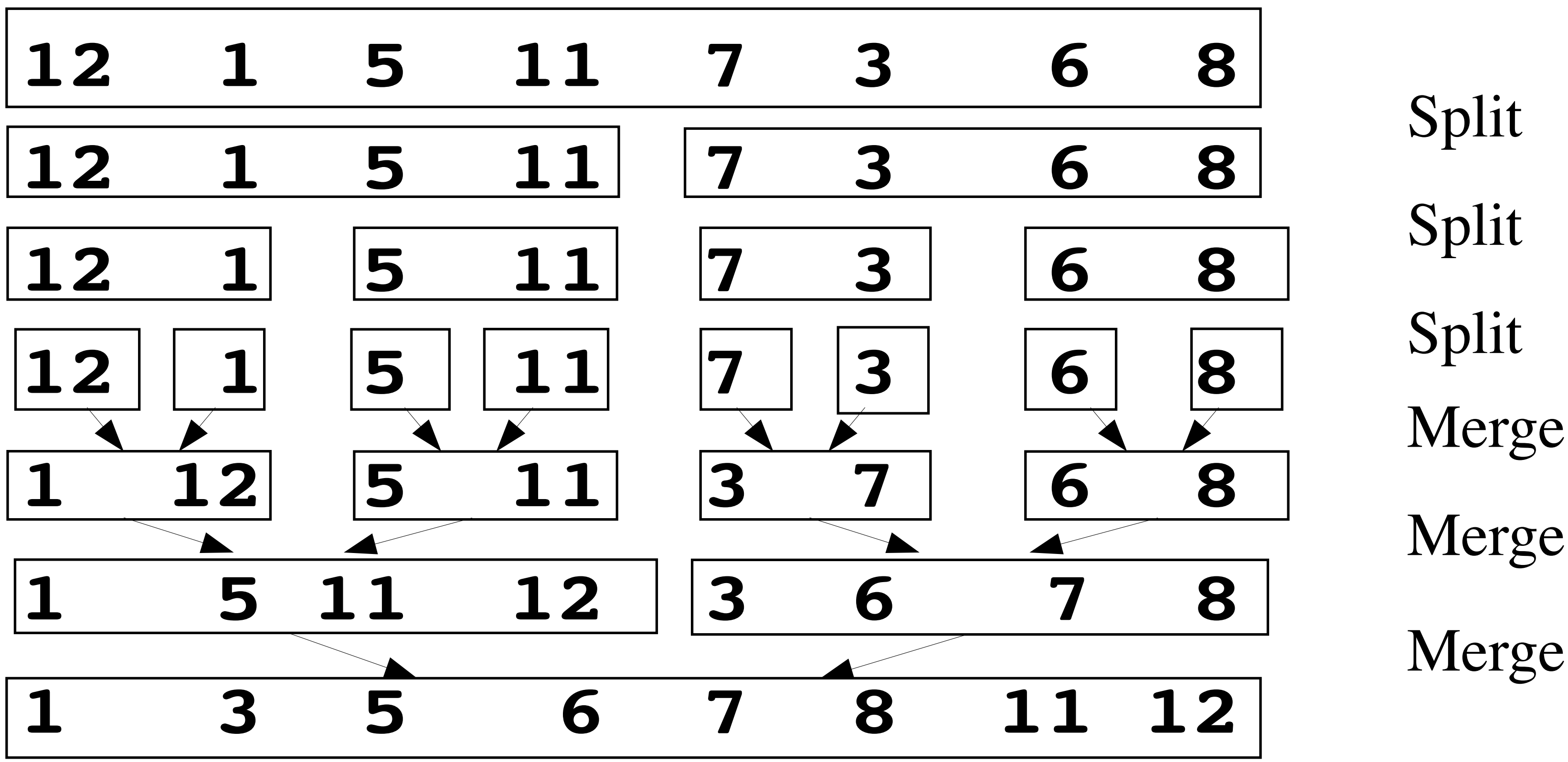
Merge *Sort*

- **Divide & Conquer:**
  - **split in two parts**
    - **no comparisons done in split**
  - **sort each part**
  - **merge the groups**
- **Cf quicksort which does comparisons in split and not in combine**

Merge Sort

Unsorted:	2	5	1	8	3	9	4
Partition:	2	5	1	8	3	9	4
Sort Groups:	1	2	5	3	4	8	9
Merge Groups:	1	2	3	4	5	8	9

Merge Sort


12 1 5 11 7 3 6 8
12 1 5 11
7 3 6 8
12 1
5 11
7 3
6 8
12
1
5
11
7
3
6
8
1 12
5 11
3 7
6 8
1 5 11 12
3 6 7 8
1 3 5 6 7 8 11 12
Split
Split
Split
Merge
Merge
Merge

Complexity

- **Merge takes O(n) where n is size of result**
- **Like quicksort, level i does 2^i sublists, each of length $O(N/2^i)$ => O(N) work at each level**
- **Best, worst, average all do O(log n) levels**
- **Complexity is O(n log n)**

Merge vs Quick

- **Merge has space overhead and also time overhead but even worst case is O(n log n)**
- **Quick is in-place and low time overhead but (very unlikely) worst case $O(n^2)$**

Topics for Final Exam

Everything from exams 1 and 2

Heaps

Graphs

Sorting

Topics for Final Exam

heaps

- heap structure and order, implementation as an array

- heap insert / sift up and delete / sift down

graphs

- directed/undirected, weighted, path, cycle, connected

- representation as adjacency matrix and adjacency list

- Depth First Search, Breadth First Search

- DFS and BFS Topsort

- Dijkstsra's shortest path algorithm

Sorting

- quicksort, heapsort, (treesort), [merge, insertion sorts]

Topics for exam 2

Binary search trees

AVL trees

Huffman codes

Hashing

Binary search trees

Ordering, Search, Insertion, Deletion,

Depth as a function of number of nodes

AVL trees

Balance factor, Rotation operation, Insertion and

rebalancing [NOT deletion], Big-O

Huffman codes

Varying length codes, Huffman trees,

Decoding, Encoding, Building the tree

Hashing

Insertion, Chaining, Searching,

Load factor and rehashing, Big-O:

search and insert, worst and expected

Topics for Exam 1

- Linked Lists
- Exceptions
- Generics
- Stacks
- Queues
- Search

Linked lists

- **add-front, delete-front, add-after, delete-after, length, last, etc**
- **circular linked lists, doubly linked lists**

Exceptions

Generics

Stacks, queues

- **as linked lists, as ArrayLists**
- **big-O**

Search

- **sequential, best/worst/average big-O**
- **binary**