SORTING

CS062

DATA STRUCTURES AND ADVANCED PROGRAMMING

18: Mergesort

Alexandra Papoutsaki

LECTURES

Mark Kampe

LABS

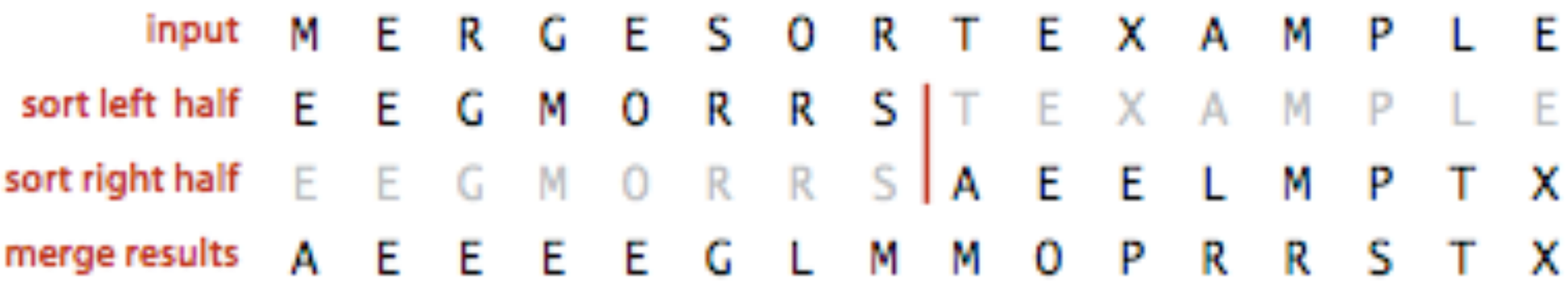
Lecture 18: Mergesort

- Mergesort

Some slides adopted from Algorithms 4th Edition or COS226

Basics

- Divide array into two halves.
- Recursively sort each half.
- Merge the two halves



Mergesort overview

http://algs4.cs.princeton.edu

2.2 MERGING DEMO

Merging

```
private static void merge(Comparable[] a, Comparable[] aux, int lo, int mid, int hi) {
    for (int k = lo; k <= hi; k++)
        aux[k] = a[k];

    int i = lo, j = mid+1;
    for (int k = lo; k <= hi; k++) {
        if (i > mid) //ran out of elements in the left subarray
            a[k] = aux[j++];
        else if (j > hi) //ran out of elements in the right subarray
            a[k] = aux[i++];
        else if (less(aux[j], aux[i]))
            a[k] = aux[j++];
        else
            a[k] = aux[i++];
    }
}
```

Merging Example - copying to auxiliary array

Array aux

A	G	L	O	R	H	I	M	S	T
0	1	2	3	4	5	6	7	8	9

Array a

A	G	L	O	R	H	I	M	S	T
0	1	2	3	4	5	6	7	8	9

lo mid hi

Merging Example - k=0

Array aux

A	G	L	O	R	H	I	M	S	T
0	1	2	3	4	5	6	7	8	9
lo				mid					hi
i				j					
k									

Array a

A	G	L	O	R	H	I	M	S	T
0	1	2	3	4	5	6	7	8	9

case: aux[i]<aux[j]

a[0]=aux[0]

i++;

Merging Example - k=1

Array aux

A	G	L	O	R	H	I	M	S	T
0	1	2	3	4	5	6	7	8	9
lo				mid					hi
i				j					
k									

Array a

A	G	L	O	R	H	I	M	S	T
0	1	2	3	4	5	6	7	8	9

case: aux[i]<aux[j]

a[1]=aux[1]

i++;

Merging Example - k=2

Array aux

A	G	L	O	R	H	I	M	S	T
0	1	2	3	4	5	6	7	8	9
lo				mid					hi
	i				j				
		k							

Array a

A	G	H	O	R	H	I	M	S	T
0	1	2	3	4	5	6	7	8	9

case: aux[i]>aux[j]

a[2]=aux[5]

j++;

Merging Example - k=3

Array aux

A	G	L	O	R	H	I	M	S	T
0	1	2	3	4	5	6	7	8	9
lo				mid					hi
	i				j				
				k					

Array a

A	G	H	I	R	H	I	M	S	T
0	1	2	3	4	5	6	7	8	9

case: aux[i]>aux[j]

a[3]=aux[6]

j++;

Merging Example - k=4

Array aux

A	G	L	O	R	H	I	M	S	T
0	1	2	3	4	5	6	7	8	9
lo				mid					hi
	i				j				
				k					

Array a

A	G	H	I	L	H	I	M	S	T
0	1	2	3	4	5	6	7	8	9

case: aux[i]<aux[j]

a[4]=aux[2]

i++;

Merging Example - k=5

Array aux

A	G	L	O	R	H	I	M	S	T
0	1	2	3	4	5	6	7	8	9
lo				mid					hi
		i				j			
				k					

Array a

A	G	H	I	L	M	I	M	S	T
0	1	2	3	4	5	6	7	8	9

case: aux[i]>aux[j]

a[5]=aux[7]

j++;

Merging Example - k=6

Array aux

A	G	L	O	R	H	I	M	S	T
0	1	2	3	4	5	6	7	8	9
lo				mid					hi
		i						j	
						k			

Array a

A	G	H	I	L	M	O	M	S	T
0	1	2	3	4	5	6	7	8	9

case: aux[i]<aux[j]

a[6]=aux[3]

i++;

Merging Example - k=7

Array aux

A	G	L	O	R	H	I	M	S	T
0	1	2	3	4	5	6	7	8	9
lo				mid					hi
		i					j		
									k

Array a

A	G	H	I	L	M	O	R	S	T
0	1	2	3	4	5	6	7	8	9

case: aux[i]<aux[j]

a[7]=aux[4]

i++;

Merging Example - k=8

Array aux

A	G	L	O	R	H	I	M	S	T
0	1	2	3	4	5	6	7	8	9
lo				mid					hi
					i			j	
									k

Array a

A	G	H	I	L	M	O	R	S	T
0	1	2	3	4	5	6	7	8	9

case: i>mid

a[8]=aux[8]

j++;

Merging Example - k=9

Array aux

A	G	L	O	R	H	I	M	S	T
0	1	2	3	4	5	6	7	8	9
lo				mid					hi
					i				j
									k

Array a

A	G	H	I	L	M	O	R	S	T
0	1	2	3	4	5	6	7	8	9

case: i>mid

a[9]=aux[9]

j++;

Practice time

How many calls does `merge()` make to `less()` in order to merge two already sorted subarrays, each of length $n/2$ into a sorted array of length n?

A. ~$1/4n$ to ~$1/2n$

B. ~$1/2n$

C. ~$1/2n$ to n

D. ~n

Answer

How many calls does `merge()` make to `less()` in order to merge two already sorted subarrays, each of length $n/2$ into a sorted array of length n?

C. ~$1/2n$ to n

That is $O(n)$

Mergesort - the quintessential example of divide-and-conquer

```
private static void sort(Comparable[] a, Comparable[] aux, int lo, int hi) {
    if (hi <= lo)
        return;
    int mid = lo + (hi - lo) / 2;
    sort(a, aux, lo, mid);
    sort(a, aux, mid+1, hi);
    merge(a, aux, lo, mid, hi);
}

public static void sort(Comparable[] a) {
    Comparable[] aux = new Comparable[a.length];
    sort(a, aux, 0, a.length - 1);
}
```

Example: Look at board for a simpler case

			a[]													
	lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
			M	E	R	G	E	S	O	R	T	E	X	A	M	P	L	E
merge(a, 0, 0, 1)			E	M	R	G	E	S	O	R	T	E	X	A	M	P	L	E
merge(a, 2, 2, 3)			E	M	G	R	E	S	O	R	T	E	X	A	M	P	L	E
merge(a, 0, 1, 3)			E	G	M	R	E	S	O	R	T	E	X	A	M	P	L	E
merge(a, 4, 4, 5)			E	G	M	R	E	S	O	R	T	E	X	A	M	P	L	E
merge(a, 6, 6, 7)			E	G	M	R	E	S	O	R	T	E	X	A	M	P	L	E
merge(a, 4, 5, 7)			E	G	M	R	E	O	R	S	T	E	X	A	M	P	L	E
merge(a, 0, 3, 7)			E	E	G	M	O	R	R	S	T	E	X	A	M	P	L	E
merge(a, 8, 8, 9)			E	E	G	M	O	R	R	S	E	T	X	A	M	P	L	E
merge(a, 10, 10, 11)			E	E	G	M	O	R	R	S	E	T	A	X	M	P	L	E
merge(a, 8, 9, 11)			E	E	G	M	O	R	R	S	A	E	T	X	M	P	L	E
merge(a, 12, 12, 13)			E	E	G	M	O	R	R	S	A	E	T	X	M	P	L	E
merge(a, 14, 14, 15)			E	E	G	M	O	R	R	S	A	E	T	X	M	P	E	L
merge(a, 12, 13, 15)			E	E	G	M	O	R	R	S	A	E	T	X	E	L	M	P
merge(a, 8, 11, 15)			E	E	G	M	O	R	R	S	A	E	E	L	M	P	T	X
merge(a, 0, 7, 15)			A	E	E	E	E	G	L	M	M	O	P	R	R	S	T	X

Trace of merge results for top-down mergesort

Practice time

Which of the following subarray lengths will occur when running mergesort on an array of length 10?

A. { 1, 2, 3, 5, 10 }
B. { 2, 4, 6, 8, 10 }
C. { 1, 2, 5, 10 }
D. { 1,2,3,4,5,10}

Answer

Which of the following subarray lengths will occur when running mergesort on an array of length 10?

A. { 1, 2, 3, 5, 10 }

Good algorithms are better than supercomputers

- Your laptop executes 10^8 comparisons per second
- A supercomputer executes 10^{12} comparisons per second

	Insertion sort			Mergesort		
Computer	**Thousand inputs**	**Million inputs**	**Billion inputs**	**Thousand inputs**	**Million inputs**	**Billion inputs**
Home	Instant	2 hours	300 years	instant	1 sec	15 min
Supercomputer	Instant	1 second	1 week	instant	instant	instant

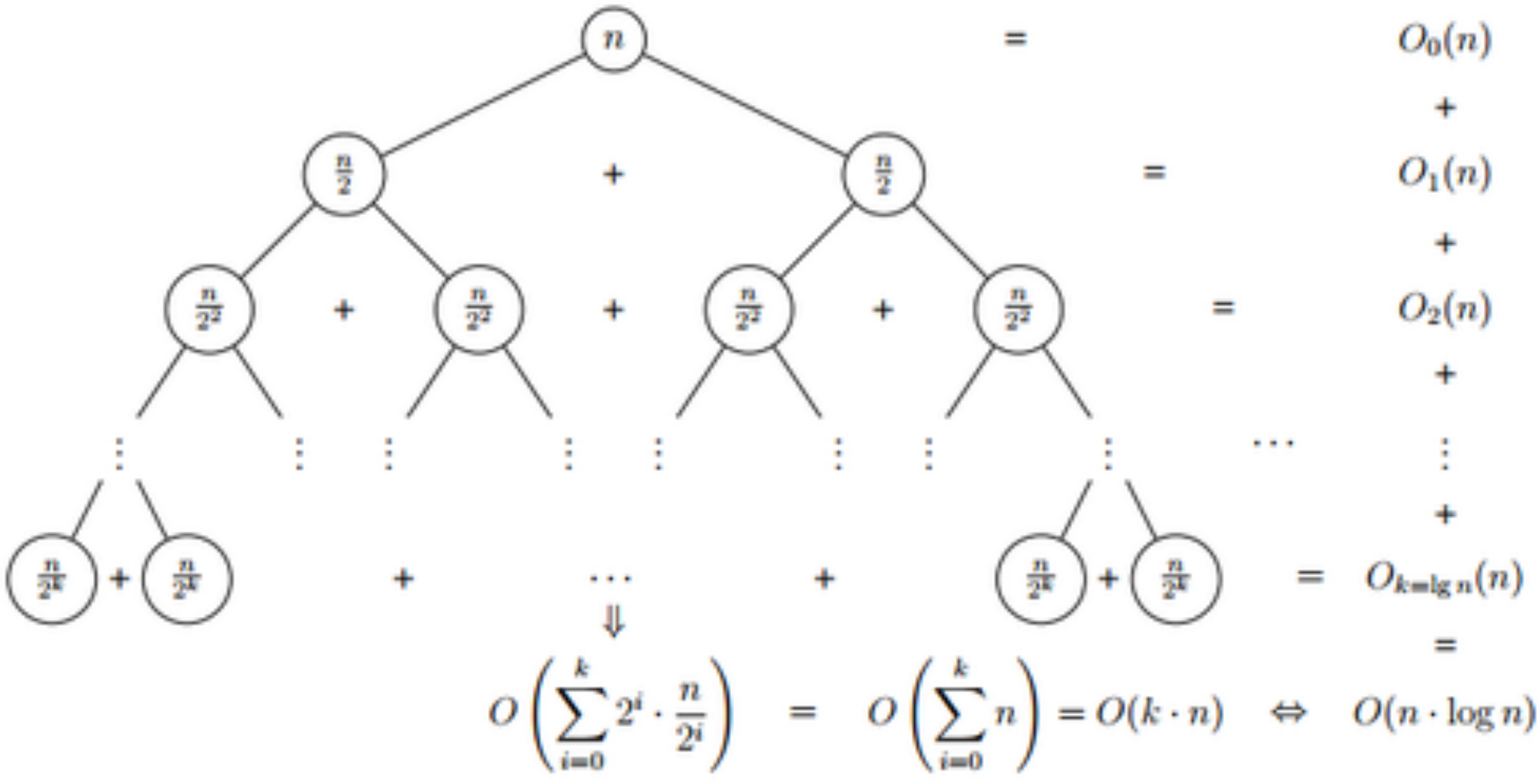
Analysis of comparisons

- The number of comparisons $C(n)$ to mergesort an array of length n satisfies the recurrence:
- $C(n) <= C(n/2) + C(n/2) + n - 1$ for $n > 1$, with $C(1) = 0$

- We will simplify the problem by assuming that n is a power of 2 and solve this simple recurrence:
- $T(n) = 2T(n/2) + n$

MERGESORT

Mergesort uses $\leq n \log n$ compares to sort an array of length n



https://cs.stackexchange.com/questions/64060/mergesort-recurssion-tree-depth-logs

Analysis of array accesses

- As with number of comparisons, mergesort uses $\leq n \log n$ array accesses to sort an array of size n.

Any algorithm with the same structure takes $n \log n$ time

```
public static void f(int n) {
     if (n == 0)
         return;
     f(n/2);
     linear(n);
}
```

Mergesort analysis of memory

- Auxiliary memory is proportional to n, as aux[] needs to be of length n for the last merge.
- At its simplest form merge sort is not an in-place algorithm.
- There are modifications for halting the size of the auxiliary array but in-place merge is very hard.

Stability

- A sorting algorithm is stable if it preserves the relative order of equal keys. For example:
  - Input: $CA_1BA_2A_3$
  - Output: $A_3A_1A_2BC$
- This sorting algorithm is not stable

Mergesort is stable

- Look into `merge()`, if equal keys it takes them from the left subarray.
- So is insertion sort, but not selection sort.

Mergesort practical improvements

- Use insertion sort for small subarrays.
  - Too much overhead for tiny subarrays.
  - Cutoff to insertion sort usually around 10 items.
  - Improvement of 10-15% in running time.
- Stop if already sorted.
  - Is largest item in first half smaller or equal the smallest item in second half? In sort method
  - `if (!less(a[mid+1], a[mid])) return;`
- Eliminate the copy the auxiliary array by saving time (not space).
  - ```
    sort (aux, a, lo, mid);
    sort (aux, a, mid+1, hi);
    merge(a, aux, lo, mid, hi);
    ```
- Java's default sort algorithm is a mergesort with all the above tricks with cutoff at 7.
    ```

Sorting: the story so far

	In place	Stable	Best	Average	Worst	Remarks
Selection	X		$1/2n^2$	$1/2n^2$	$1/2n^2$	n exchanges
Insertion	X	X	n	$1/4n^2$	$1/2n^2$	Use for small arrays or partially ordered
Merge sort		X	$1/2n \log n$	$n \log n$	$n \log n$	Guaranteed performance; stable

Lecture 18: Mergesort

- Mergesort

Readings:

- Textbook:
  - Chapter 2.2 (pages 270–277)
- Website:
  - Mergesort: https://algs4.cs.princeton.edu/22mergesort/
  - Code: https://algs4.cs.princeton.edu/22mergesort/Merge.java.html

Practice Problems:

- 2.2.1-2.2.2, 2.2.11