




Sorting




Announcements

- DNA - Due March 5
- APT Set 5 - Due ?
- Spring Break!!!!!!



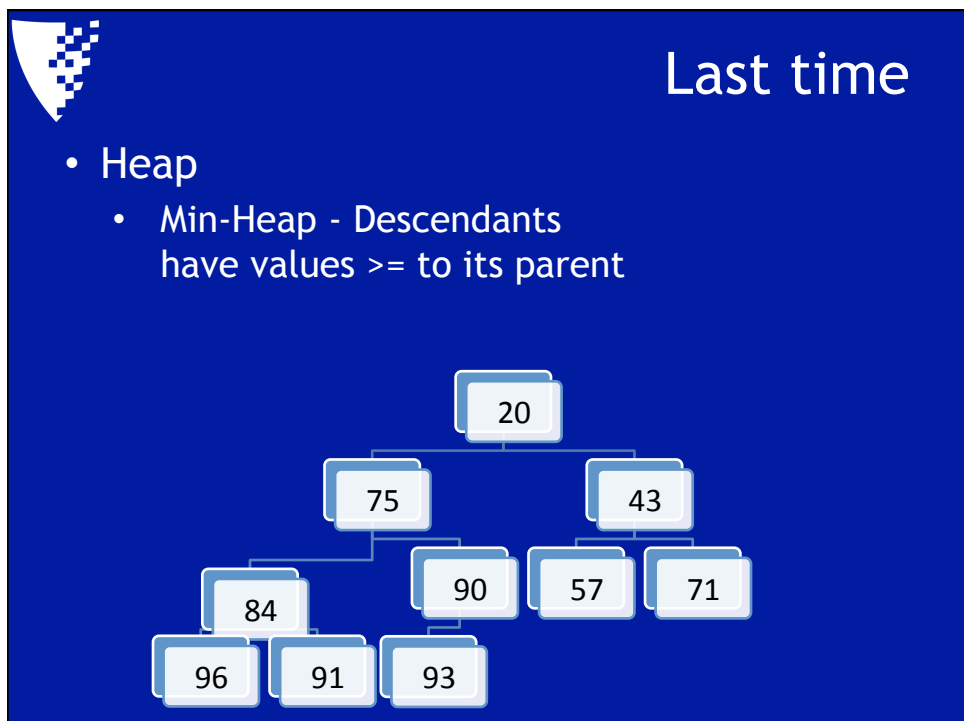
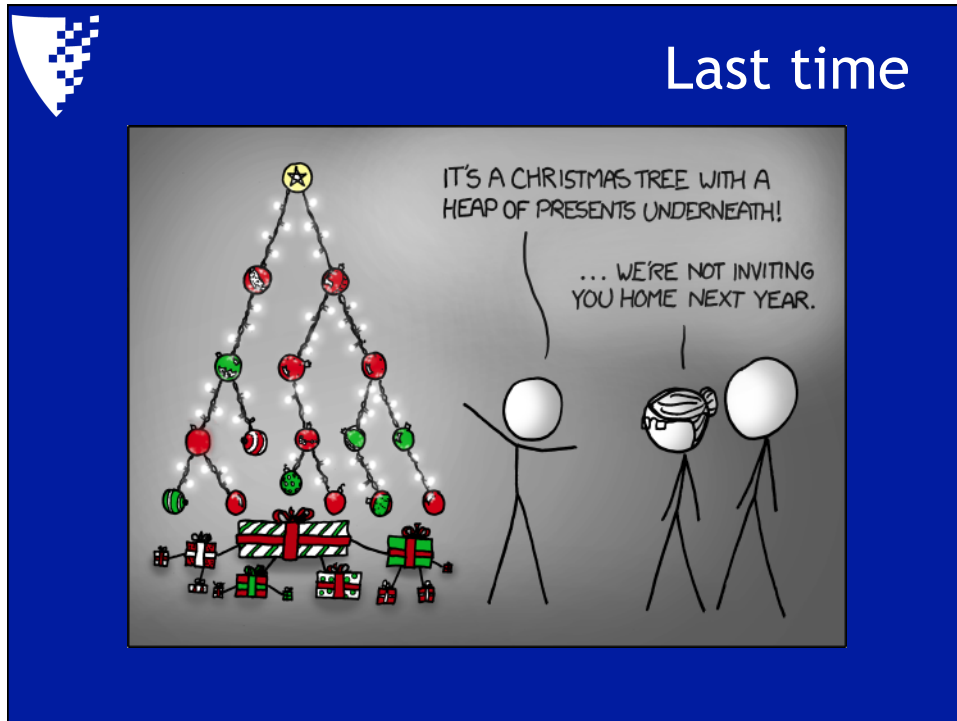
March

March					
	Monday	Tuesday	Wednesday	Thursday	Friday
	25	26	27	28	1
Trees, Sorting, Priority Queues	4	5 DNA Due	6	7	8
	11 Spring	12 Break	13 Woo	14 !!!!!!!	15 !!!!!!!
Backtracking	18	19	20	21 20 Questions Due	22
Review	25 APT Set 5 Due	26	27 Exam 2	28	29



APTs

- Word Ladder



Priority Queues

- A queue with priority
- Highest priority removed first
 - Priority determined with compareTo
 - numbers - lowest first
 - words - alphabetical

```

graph TD
    20 --> 75
    20 --> 43
    75 --> 84
    75 --> 90
    84 --> 96
    84 --> 91
    90 --> 93
    43 --> 57
    43 --> 71
  
```

Priority Queues

- If `q` is a priority queue

```

int[] array = new int[q.size()];
int i = 0;
while(!q.isEmpty()){
    array[i] = q.remove();
    i++;
}
return array;
  
```

- What is in array?
- What is the running time?
 - What is the running time of `q.remove()`?

Sorting


Sorting Algorithm	Best	Average	Worst
Heapsort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$

Sorting



The slide contains four images illustrating sorting concepts:

- Top-left: A pile of multi-colored candies (M&M's) next to several colored plastic cups, representing an unsorted state.
- Top-right: Hands sorting small white cards on a wooden table, representing a manual sorting process.
- Bottom-left: A hand pointing to a page in an open book, representing a sorted list or index.
- Bottom-right: A pile of M&M's candies sorted by color into distinct groups, representing a sorted state.



Animation

Merge Sort


```

public void sort(int[] array){
    if(array.length > 1){
        int half = array.length/2;
        int[] a1 = Arrays.copyOfRange(array, 0, half);
        int[] a2 = Arrays.copyOfRange(array, half, array.length);
        sort(a1);
        sort(a2);
        merge(array, a1, a2);
    }
}

private void merge(int[] array, int[] a1, int[] a2){
    int len1 = a1.length; int it1 = 0;
    int len2 = a2.length; int it2 = 0;

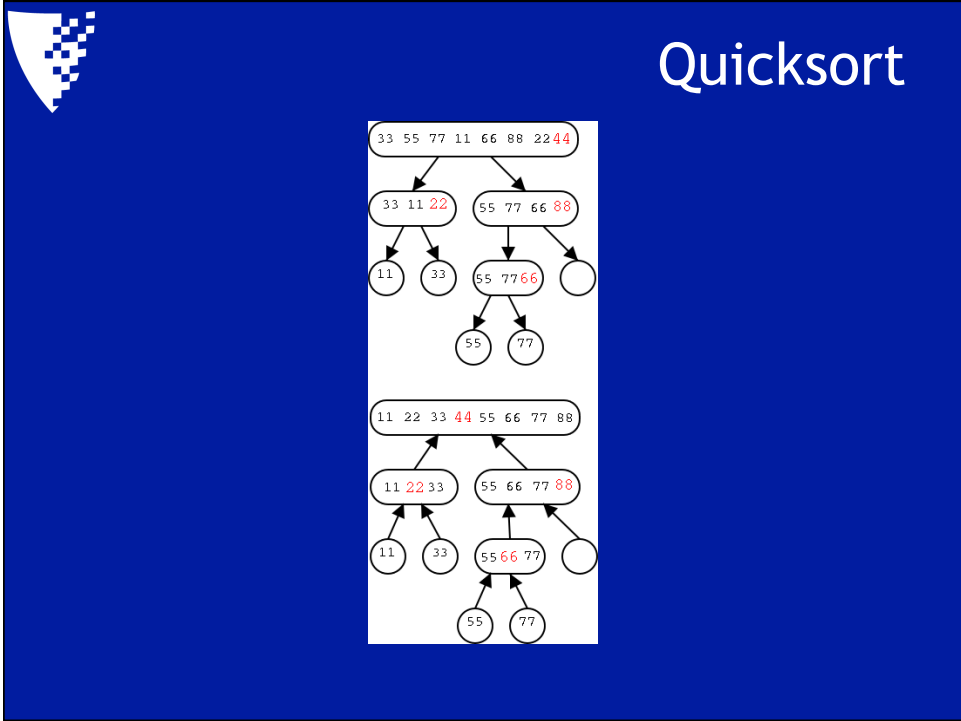
    for(int i=0; i < array.length; i++){
        if(it2==len2 || (it1 < len1 && a1[it1] < a2[it2])){
            array[i] = a1[it1];
            it1++;
        }
        else{
            array[i] = a2[it2];
            it2++;
        }
    }
}

```



Sorting

Sorting Algorithm	Best	Average	Worst
Heapsort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
Mergesort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$



Sorting

Sorting Algorithm	Best	Average	Worst
Heapsort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
Mergesort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
Quicksort	$O(n \log n)$	$O(n \log n)$	$O(n^2)$



 A slide with a dark blue background. In the top-left corner, there is a white logo consisting of a square with a grid of dots, some of which are missing, creating a pixelated effect. The title "Sorting" is written in white, sans-serif font in the top-right corner. Below the title is a table with a yellow header row and four data rows. The table lists sorting algorithms and their time complexities in the best, average, and worst cases.


Sorting Algorithm	Best	Average	Worst
Heapsort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
Mergesort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
Quicksort	$O(n \log n)$	$O(n \log n)$	$O(n^2)$
Bubblesort	$O(n)$	$O(n^2)$	$O(n^2)$

Insertion Sort




Sorting

Sorting Algorithm	Best	Average	Worst
Heapsort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
Mergesort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
Quicksort	$O(n \log n)$	$O(n \log n)$	$O(n^2)$
Bubblesort	$O(n)$	$O(n^2)$	$O(n^2)$
Insertionsort	$O(n)$	$O(n^2)$	$O(n^2)$



Heap sort

Sorting Algorithm	Best	Average	Worst
Heapsort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
Mergesort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
Quicksort	$O(n \log n)$	$O(n \log n)$	$O(n^2)$
Bubblesort	$O(n)$	$O(n^2)$	$O(n^2)$
Insertionsort	$O(n)$	$O(n^2)$	$O(n^2)$
Bogosort	$O(n)$	$n * n!$	∞



Take home

- There are a lot of sorting algorithms
- Some are better than others
- $O(n \log n)$ is good!
- Do NOT write your own sorting algorithm
 - It has been done for you



March

	March				
	Monday	Tuesday	Wednesday	Thursday	Friday
	25	26	27	28	1
Trees, Sorting, Priority Queues	4	5 DNA Due	6	7	8
	11 Spring	12 Break	13 Woo	14 !!!!!!!	15 !!!!!!!
Backtracking	18	19	20	21 20 Questions Due	22
Review	25 APT Set 5 Due	26	27 Exam 2	28	29