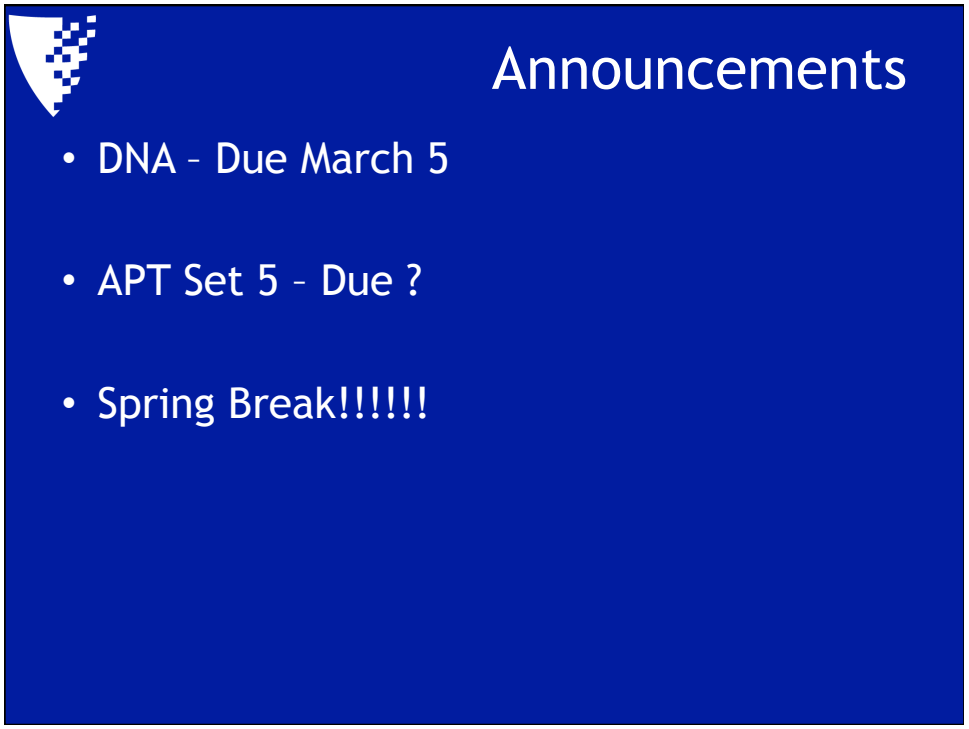
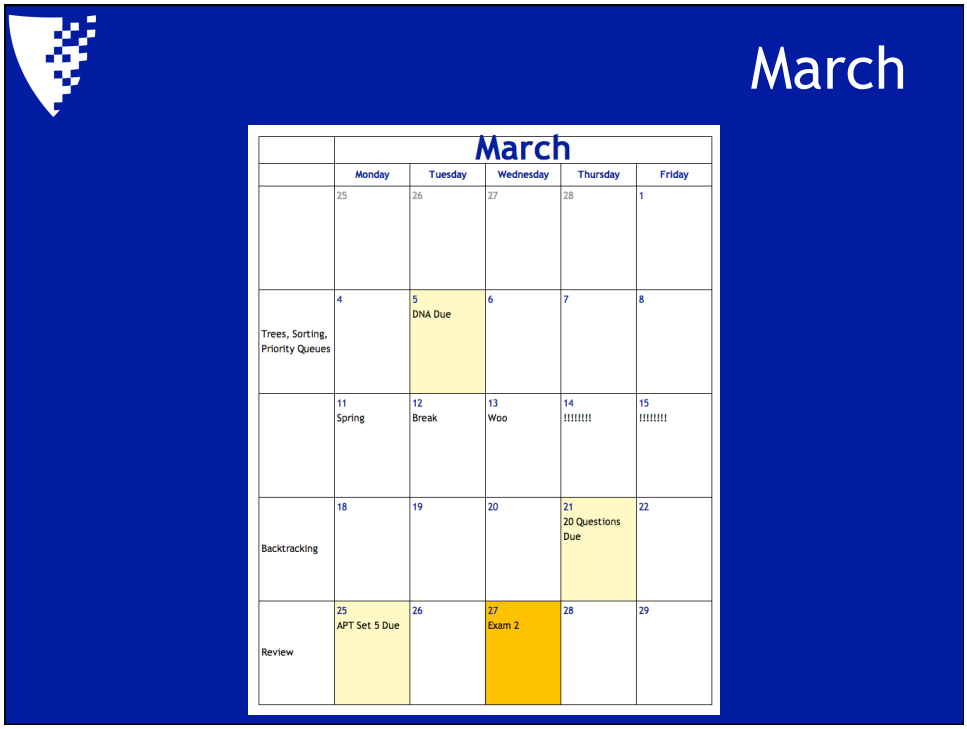
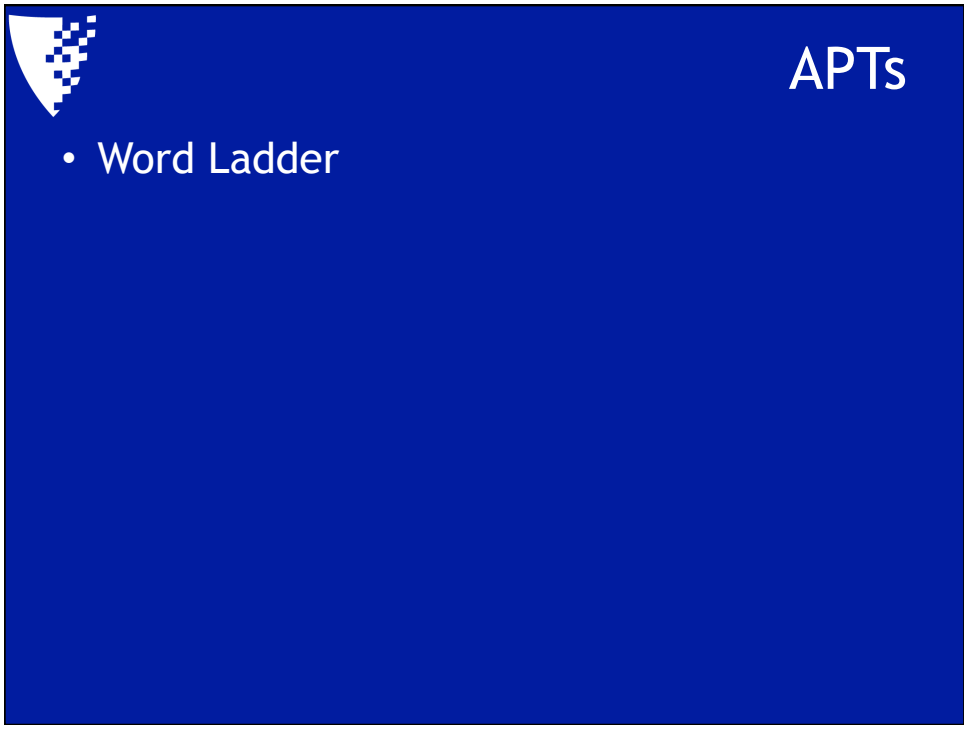

Sorting


Announcements
• DNA – Due March 5
• APT Set 5 – Due ?
• Spring Break!!!!!!



March

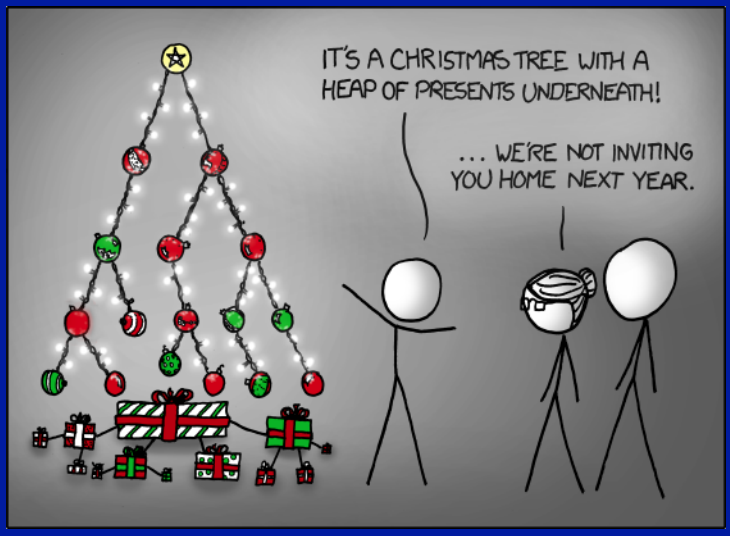
	March				
	Monday	Tuesday	Wednesday	Thursday	Friday
	25	26	27	28	1
Trees, Sorting, Priority Queues	4	5 DNA Due	6	7	8
	11 Spring	12 Break	13 Woo	14 !!!!!!!!	15 !!!!!!!!
Backtracking	18	19	20	21 20 Questions Due	22
Review	25 APT Set 5 Due	26	27 Exam 2	28	29



APTs

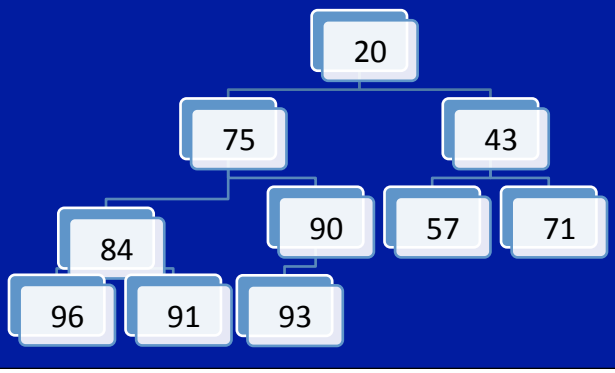
- Word Ladder

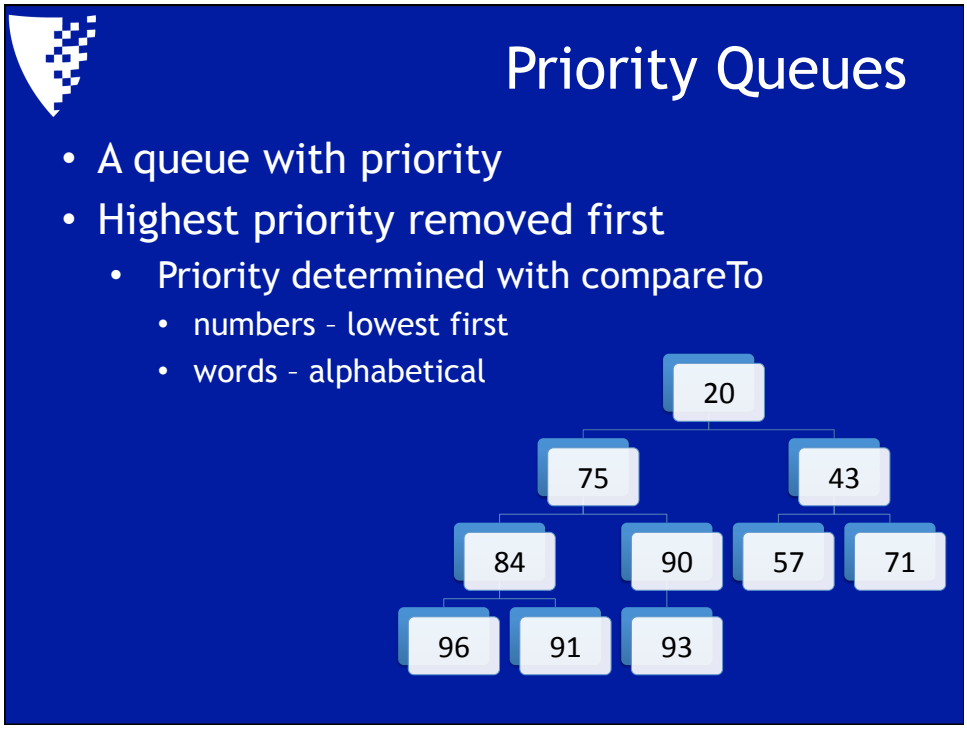
Last time

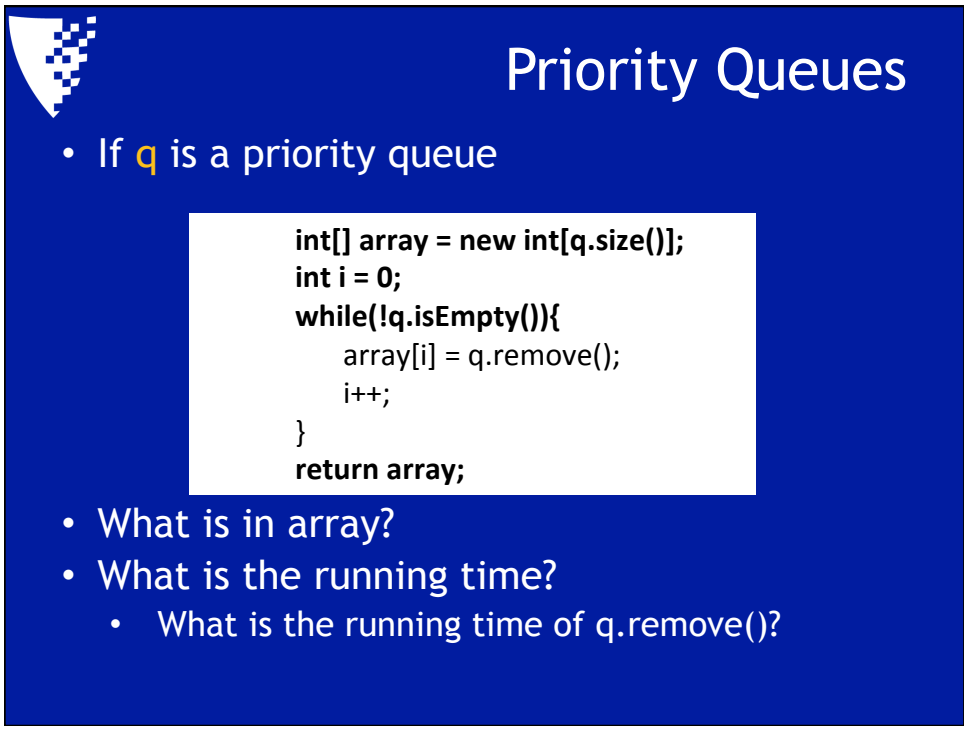


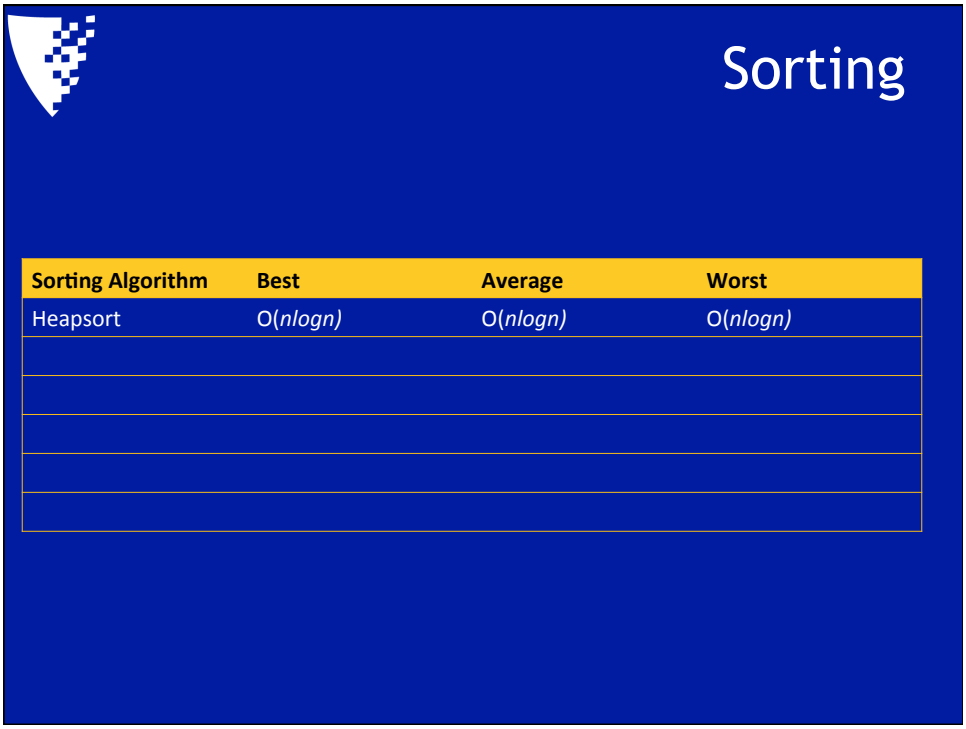
Last time

- Heap
  - Min-Heap - Descendants have values >= to its parent




Priority Queues
A queue with priority
Highest priority removed first
Priority determined with compareTo
numbers - lowest first
words - alphabetical
20
75
43
84
90
57
71
96
91
93


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(nlogn)
O(nlogn)
O(nlogn)


Sorting

Merge Sort

- Animation

```
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);
        }
    }
```

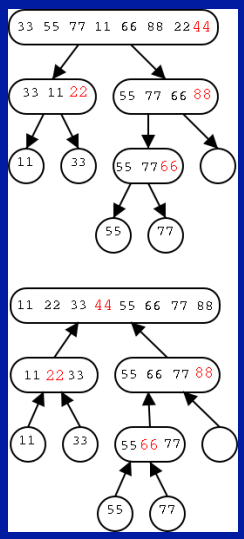
```
private void merge(int[] array, int[] a1, int[] a2){
    int len1 = a1.length; int it1 = 0;
    int len2 = a2.length; int it2 = 0;

    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(*nlogn*)	O(*nlogn*)	O(*nlogn*)
Mergesort	O(*nlogn*)	O(*nlogn*)	O(*nlogn*)

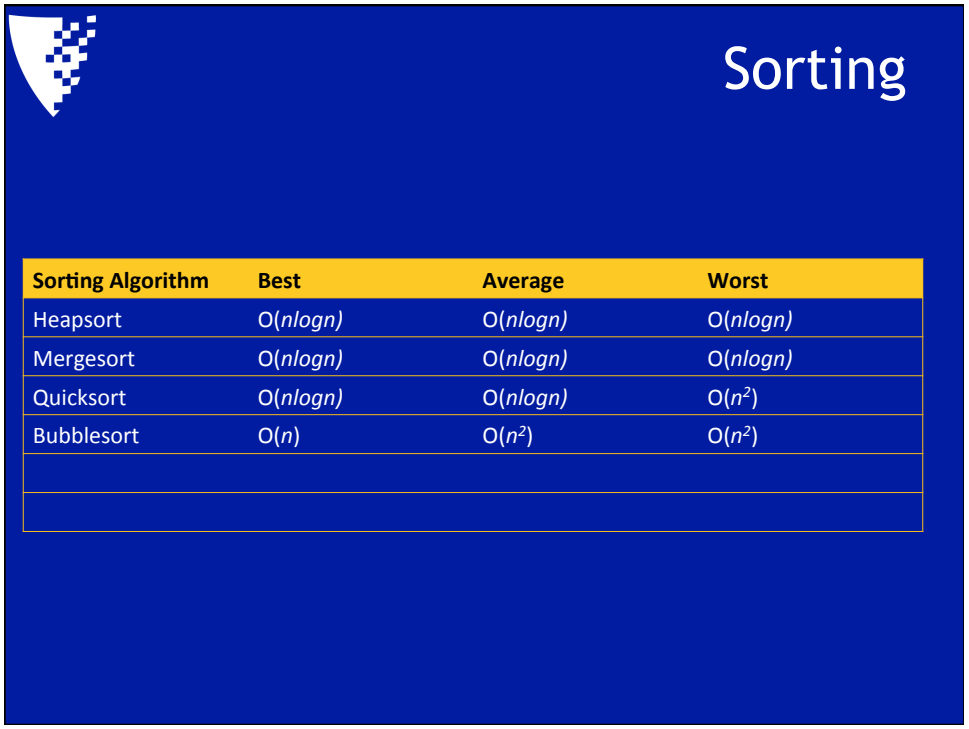
Quicksort



Sorting

Sorting Algorithm	Best	Average	Worst
Heapsort	O(*nlogn*)	O(*nlogn*)	O(*nlogn*)
Mergesort	O(*nlogn*)	O(*nlogn*)	O(*nlogn*)
Quicksort	O(*nlogn*)	O(*nlogn*)	O(n^2)

Bubble Sort



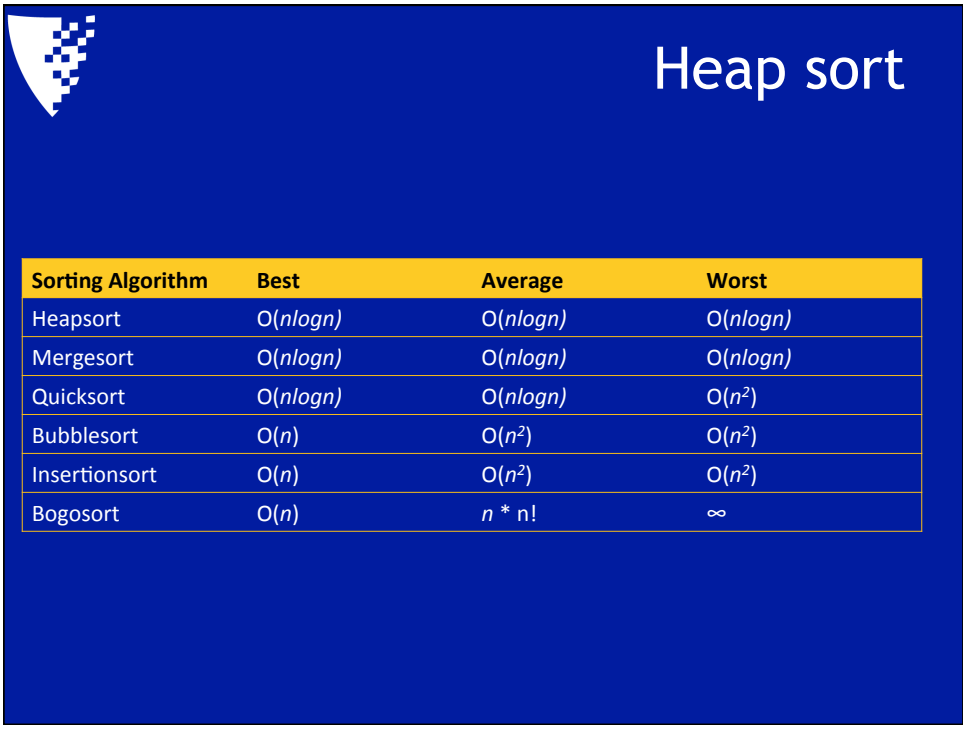
Sorting

Sorting Algorithm	Best	Average	Worst
Heapsort	O(*nlogn*)	O(*nlogn*)	O(*nlogn*)
Mergesort	O(*nlogn*)	O(*nlogn*)	O(*nlogn*)
Quicksort	O(*nlogn*)	O(*nlogn*)	O(n^2)
Bubblesort	O(*n*)	O(n^2)	O(n^2)

Insertion Sort

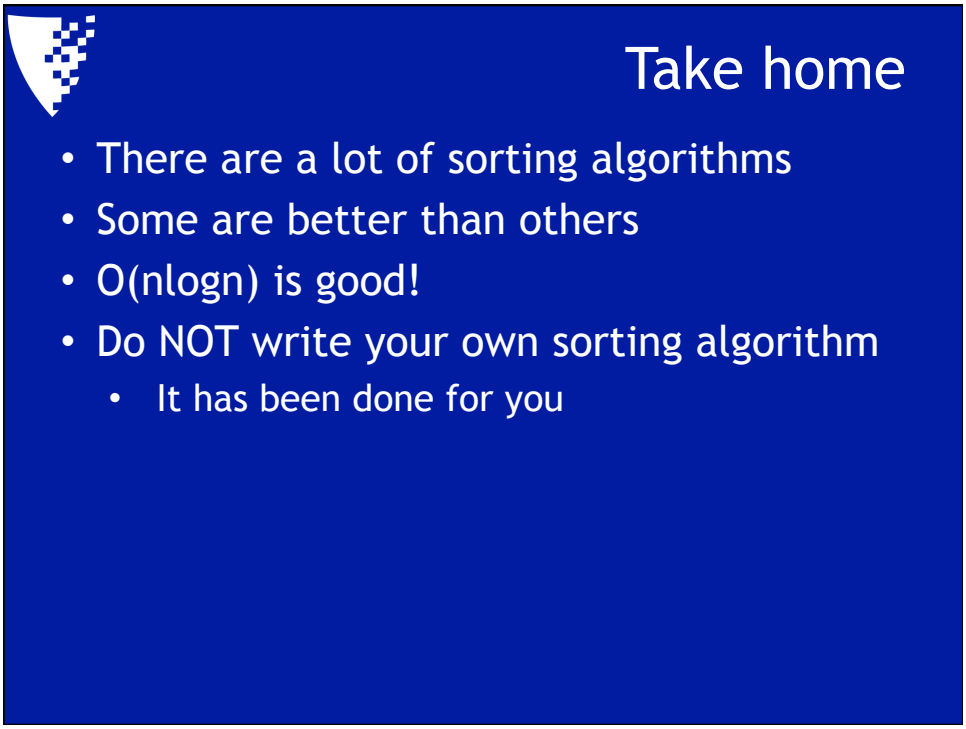
Sorting

Sorting Algorithm	Best	Average	Worst
Heapsort	O(*nlogn*)	O(*nlogn*)	O(*nlogn*)
Mergesort	O(*nlogn*)	O(*nlogn*)	O(*nlogn*)
Quicksort	O(*nlogn*)	O(*nlogn*)	O(n^2)
Bubblesort	O(*n*)	O(n^2)	O(n^2)
Insertionsort	O(*n*)	O(n^2)	O(n^2)



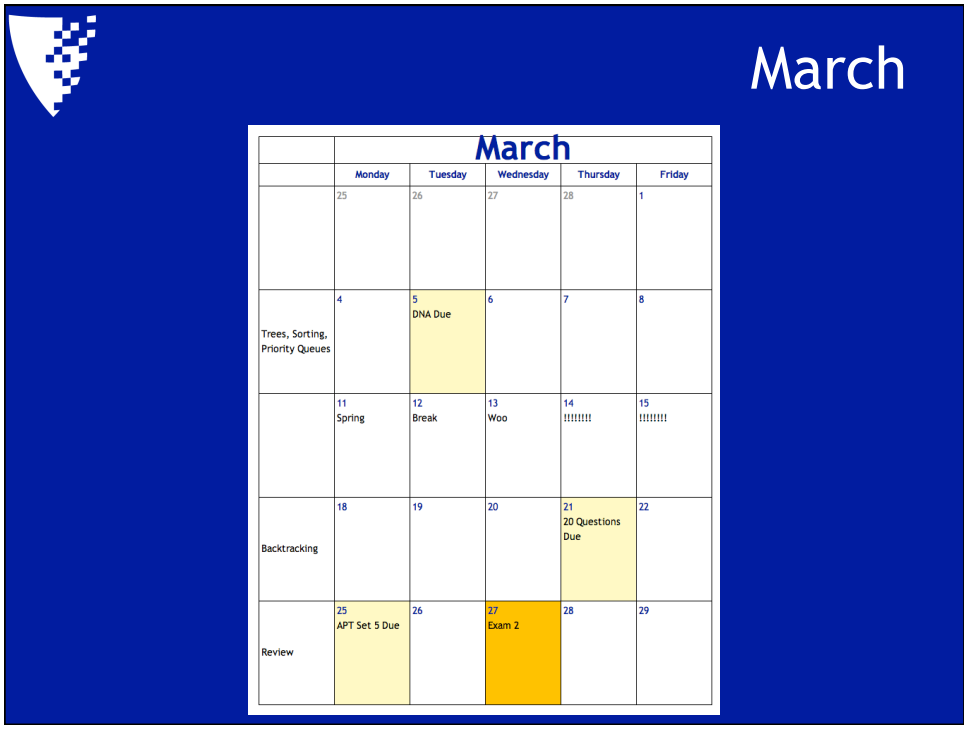
Heap sort

Sorting Algorithm	Best	Average	Worst
Heapsort	O(*nlogn*)	O(*nlogn*)	O(*nlogn*)
Mergesort	O(*nlogn*)	O(*nlogn*)	O(*nlogn*)
Quicksort	O(*nlogn*)	O(*nlogn*)	$O(n^2)$
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(nlogn) 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