

# Data structures

Exercise session – Week 2

# I. Sorting

# Sorting by *Insertion*

|4 6 8 2 9 5 1 7 3  
4 |6 8 2 9 5 1 7 3  
4 6 |8 2 9 5 1 7 3  
4 6 8 |2 9 5 1 7 3  
2 4 6 8 |9 5 1 7 3  
2 4 6 8 9 |5 1 7 3  
2 4 5 6 8 9 |1 7 3  
1 2 4 5 6 8 9 |7 3  
1 2 4 5 6 7 8 9 |3  
1 2 3 4 5 6 7 8 9|

# Sort the sequence, Quick!

~~4~~ 6 8 2 9 5 1 7 3

2 1 3 ~~4~~ 6 8 9 5 7

1 2 3 ~~4~~ 5 ~~6~~ 8 9 7

1 2 3 4 5 6 ~~7~~ 8 9

1 2 3 4 5 6 7 8 9

# sequence Merge the Sort

4 6 8 2 9 5 1 7 3

4 6 8 2---9 5 1 7 3

4 6---8 2---9 5---1 7 3

4---6---8---2---9---5---1---7 3

4---6---8---2---9---5---1---7---3

4 6---2 8---5 9---1---3 7

4 6---2 8---5 9---1 3 7

2 4 6 8---1 3 5 7 9

1 2 3 4 5 6 7 8 9

# Är du en stable sorting algorithm?

Original:

peach, *straw*, apple, *spork*

Stable:

apple, peach, *straw*, *spork*

Go home, you're unstable:

apple, peach, *spork*, *straw*

(from [StackOverflow](#))

# Exercise!

Is it possible to remove all duplicate elements from an array in  $O(n \log n)$  time? How?

# II. Complexity



# Exercise!

Print all subsets of a set  
(given as an array)

# Estimating time complexity

$$T(0) = O(1), T(n) = 2 * T(n-1)$$

$$T(1) = 2 * T(0) = 2 * O(1)$$

$$T(2) = 2 * T(1) = 4 * O(1)$$

$$T(3) = 2 * T(2) = 8 * O(1)$$

...

$$T(n) = 2^n * O(1) = O(2^n)$$

# II. Stacks & Queues

```
interface Stack<E>
```

```
    void push(E a)
```

```
    E pop()
```

# ”Last In, First out”

```
stack.push(1)
```

```
stack.push(2)
```

```
stack.pop()    // returns 2
```

```
stack.push(3)
```

```
stack.pop()    // returns 3
```

```
stack.pop()    // returns 1
```

# Stacks are cool

Used by JVM to keep track of order of function calls

f() calls g()

g() calls h()

h() calls i()

...

# Exercise'o clock!

Give an algorithm that removes all comments from a program

# Exercise'o clock!

Give an algorithm that reads a *postfix* expression and evaluates it



# Queue

```
void enqueue(E a)
```

```
E dequeue()
```

# ”First In, First out”

```
q.enqueue(1)
```

```
q.enqueue(2)
```

```
q.dequeue()           // returns 1
```

```
q.enqueue(3)
```

```
q.dequeue()          // returns 2
```

```
q.dequeue()          // returns 3
```