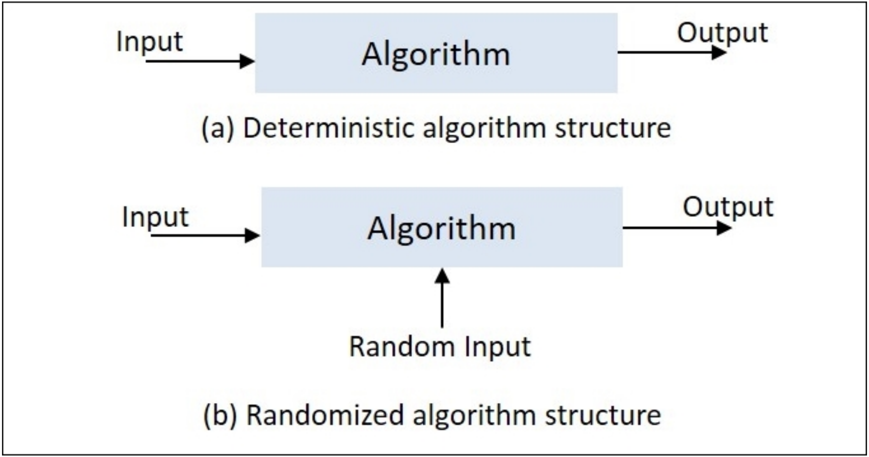
## What are...randomized algorithms?

Or: Randomness rocks!

## Algorithm + random = ?

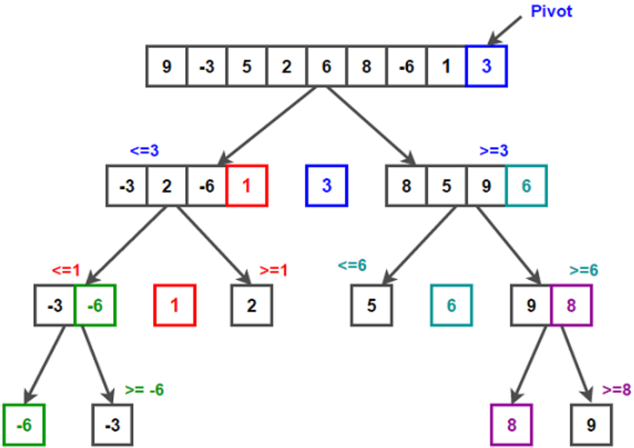



(a) Deterministic algorithm structure

(b) Randomized algorithm structure

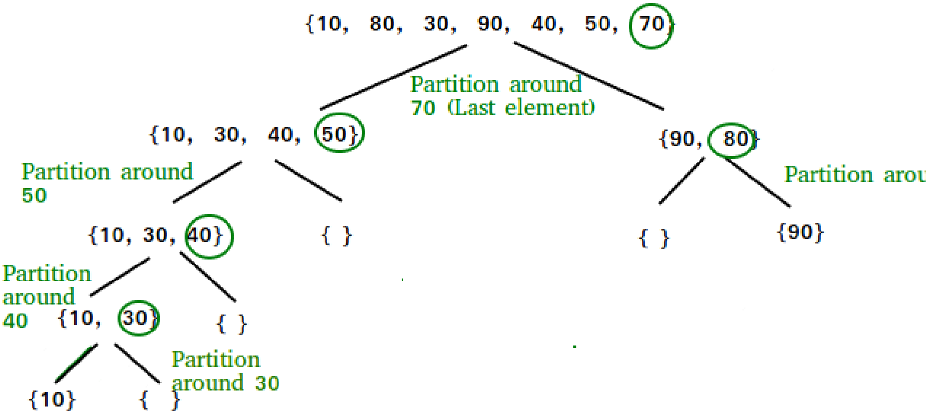
- Algorithm A deterministic procedure
- Random The opposite ;-)
- Randomized algorithm A degree of randomness in an deterministic procedure

# Quicksort




- Goal Sort a set $X$
- Take a pivot $p$ and divide into $X(< p)$ and $X(\geq p)$ Divide
- Repeat with $X(< p)$ and $X(\geq p)$ Conquer

# Pick a random pivot




- Best/worst case scenario Divide into two equally sized sets resp. as above
- Idea: Choose a pivot randomly!
- In this way one ends in the average situation most of the time

## Enter, the theorem

Take a set $X$ with $n$ elements
The expected-case resp. worst-case time of randomized quicksort is

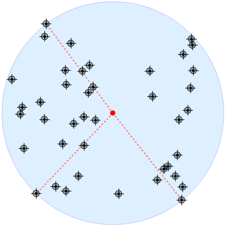
$$O(n \log n) \text{ resp. } O(n^2)$$

- The average-case resp. worst-case time of plain quicksort is

$$O(n \log n) \text{ resp. } O(n^2)$$

  $\Rightarrow$ good average-case performance but not good worst-case performance
- The difference?
  (a) Average = on random input
  (b) Expected = on every input with large probability
- Making the algorithm probabilistic gives more control over the running time
- Randomizing algorithms works way more general than for quicksort

# The smallest-circle problem



```
algorithm welzl is
    input: Finite sets P and R of points in the plane |R| ≤ 3.
    output: Minimal disk enclosing P with R on the boundary.

    if P is empty or |R| = 3 then
        return trivial(R)
    choose p in P (randomly and uniformly)
    D := welzl(P − {p}, R)
    if p is in D then
        return D

    return welzl(P − {p}, R ∪ {p})
```

- Goal Find the smallest circle surrounding given points $X$
- Take randomly $p$ and find the smallest circle for $X \setminus \{p\}$ recursively
- The expect run time is $O(n)$

**Thank you for your attention!**

---

I hope that was of some help.