11.0 - Week 11 - Workshop (MA)

Learning Objectives

- Recursion
- Recursion vs Iteration
- Recursive Sorts

Week 11 Padlet Discussion Board link: https://monashmalaysia.padlet.org/fermi/2022week11

What is Recursion?

Question *Submitted Oct 10th 2022 at 10:23:04 am*

What is called ***recursion*** in computer science?

- ○ Something like *GNU stands for "GNU's not Unix"*
- [✓ ●] It is a method of solving a problem such that its solution depends on solutions to smaller instances of ***the same problem***.
- ○ TRUE programmers use iteration only!

Recursion - Example

> *"The power of recursion evidently lies in the possibility of defining an infinite set of objects by a finite statement. In the same manner, an infinite number of computations can be described by a finite recursive program, even if this program contains no explicit repetitions."*

> — Niklaus Wirth, Algorithms + Data Structures = Programs, 1976

Recursion can be traced back to ***recurrence relations*** in mathematics:

$$u_n = \varphi(n, u_{n-1}) \text{ for } n > 0, \\ \text{s.t. } u_0 \in X \text{ and } \varphi : \mathbb{N} \times X \to X$$

Recursion is also the central concept of ***computability theory***.

```
def gcd(a: int, b: int) -> int:
    """ Euclidean algorithm to find a GCD of two integers a and b. """

    while True:
        r = a % b
        if r == 0:
            return b
        a, b = b, r

if __name__ == '__main__':
    a = int(input('Enter a: '))
    b = int(input('Enter b: '))
    print(gcd(a, b))
```

Recursion - Another example

Here is a function that enumerates ***all prime factors*** of a natural number $n \in \mathbb{N}$:

```
def fact(n: int) -> None:
    """ Number factorization. """

    k = 2

    factors = []
    while k * k <= n:
        if n % k == 0:
            factors.append(k)
            n = n // k
        else:
            k += 1

    factors.append(n)
    return factors

if __name__ == '__main__':
    n = int(input('Enter number: '))
    print(fact(n))
```

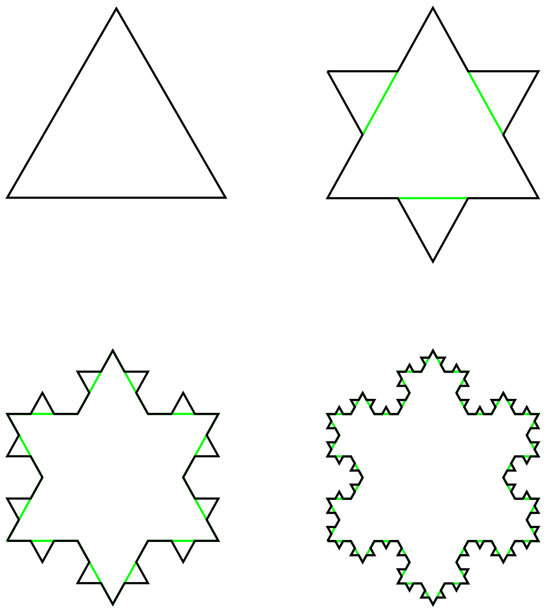
Our task is to implement its **recursive equivalent**.

A Bit of Fun

✓ Recursion is a natural way to draw **fractal curves**.

One simple example of a *fractal curve* is ***Koch snowflake***. The Koch snowflake can be constructed by starting with an equilateral triangle, then recursively altering each line segment as follows:

- divide the line segment into ***three segments*** of equal length.
- draw an equilateral triangle that has the middle segment from step 1 as its base and points outward.
- remove the line segment that is the base of the triangle from step 2.



Hint: ***Koch anti-snowflake*** also looks nice!

Recursion or Iteration?

Question 1 *Submitted Oct 10th 2022 at 10:23:16 am*

Is it recursion or iteration that is typically better at representing the code in a *clean and short* manner?

- [✓ Recursion]
- Iteration
- My code is perfect either way!

Question 2 *Submitted Oct 10th 2022 at 10:23:25 am*

Recursion stops when ...

- loop continuation condition fails.
- [✓ base case is reached.]
- ... never!

Question 3 *Submitted Oct 10th 2022 at 10:23:37 am*

What is believed to be normally slower?

- [✓ Recursion, due to the overhead of maintaining the call stack.]
- Iteration, due to initialising the necessary variables and checking the loop continuation condition.

Question 4 *Submitted Oct 10th 2022 at 10:23:44 am*

Can recursive code be transformed into iteration and/or vice versa?

○ No, it is impossible either way.

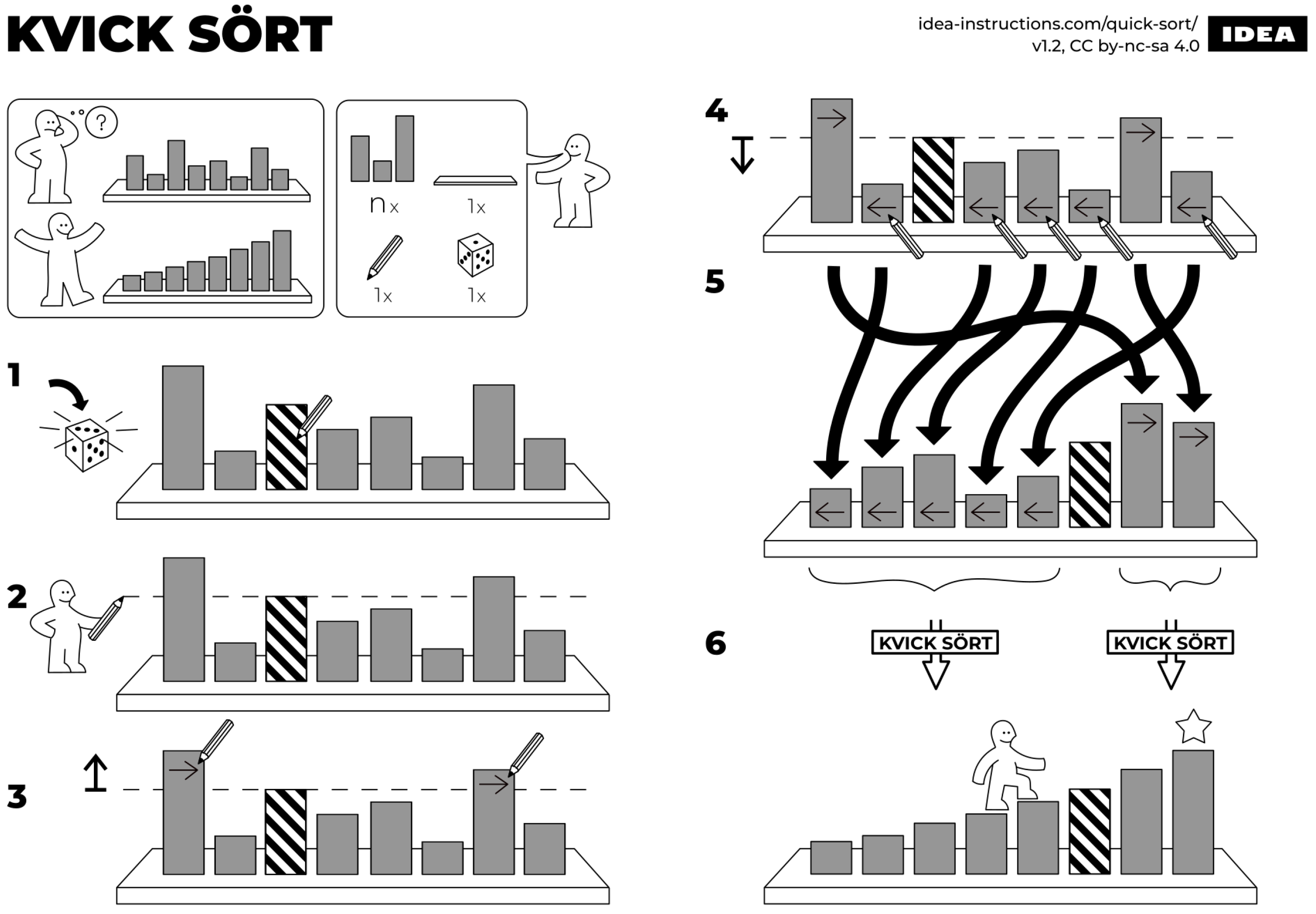
○ Yes, recursion to iteration only.

○ Yes, iteration to recursion only.

[✓ Both ways but may be tricky.]

QuickSort as Recursive Sort

✓ **Recursion can be used to sort sequences of numbers efficiently!**

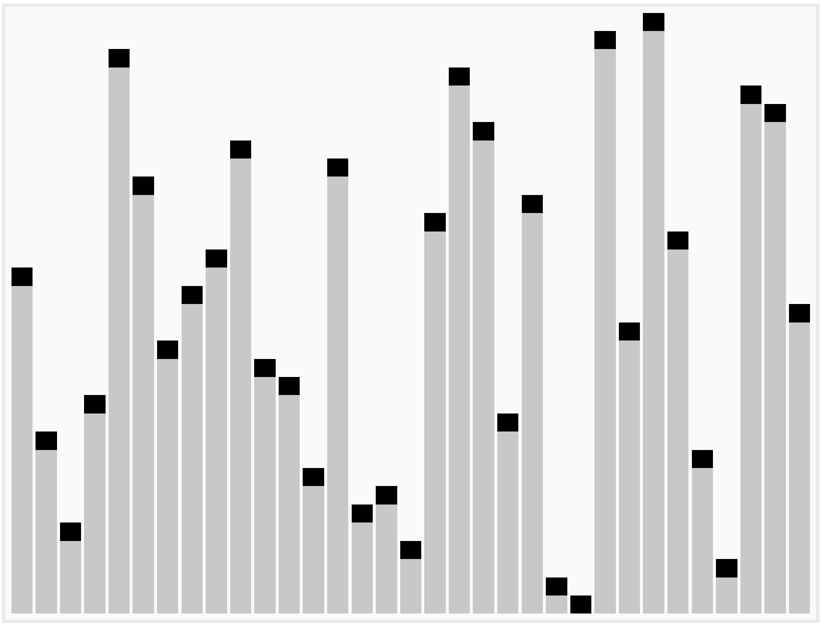


i Recursion-based algorithms are often called **divide-and-conquer** algorithms!

QuickSort Implementation

Idea:

1. pick a *pivot* element
2. move all elements larger than the pivot *to the right* and all smaller elements *to the left*
3. apply QuickSort *recursively* to the sub-arrays of smaller and larger elements



The choice of the pivot is ***crucial*** for the performance of the algorithm!

QuickSort was invented by **Tony Hoare** in 1959.

Complexity of QuickSort

Question 1 *Submitted Oct 11th 2022 at 8:02:58 am*

What is the *best-case complexity* of QuickSort assuming the array size is n?

- ○ $\mathcal{O}(1)$
- [✓ $\mathcal{O}(n \times \log n)$]
- ○ $\mathcal{O}(n)$
- ○ $\mathcal{O}(n^2)$

Question 2 *Submitted Oct 11th 2022 at 8:03:08 am*

What is the *worst-case complexity* of QuickSort assuming the array size is n?

- ○ $\mathcal{O}(1)$
- ○ $\mathcal{O}(n \times \log n)$
- ○ $\mathcal{O}(n)$
- [✓ $\mathcal{O}(n^2)$]

Question 3 *Submitted Oct 11th 2022 at 8:03:19 am*

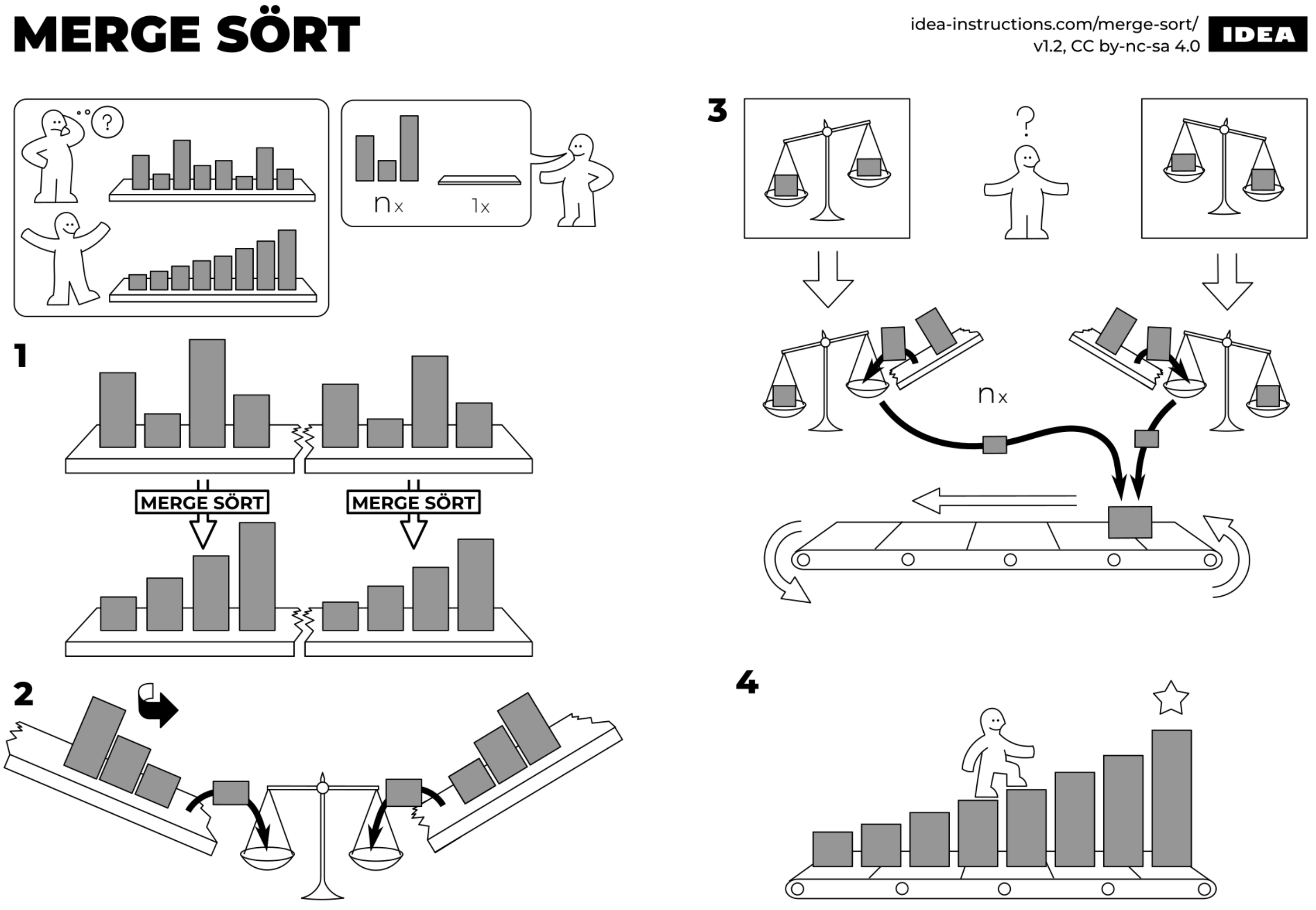
Wait, why is it ***"quick"*** then?!

- ○ Well, ***our lecturers*** *insist on this* and we trust them!
- ○ Because other sorting algorithms are even worse than that! 😟

● Because its *average complexity* is known to be $\mathcal{O}(n \times \log n)$.

MergeSort

✓ **MergeSort** is another example of a sorting algorithm, which uses **recursion**.



MergeSort Implementation

Idea:

1. *split* the array into sub-arrays, ideally of equal size
2. sort each sub-array recursively
3. apply a *merging procedure* to the sorted sub-arrays to get the full array sorted

MergeSort was invented by **John von Neumann** in 1945.

Complexity of MergeSort

Question 1 *Submitted Oct 11th 2022 at 8:04:21 am*

What is the *worst-case complexity* of `merge()` assuming the sizes of the sub-arrays to merge are m and l, such that $m + l = n$?

- ○ $\mathcal{O}(1)$
- ○ $\mathcal{O}(n \times \log n)$
- [✓ $\mathcal{O}(n)$]
- ○ $\mathcal{O}(n^2)$
- ○ $\mathcal{O}(m)$
- ○ $\mathcal{O}(l)$
- ○ $\mathcal{O}(m \times \log l)$

Question 2 *Submitted Oct 11th 2022 at 8:04:27 am*

What is the *worst-case complexity* of MergeSort assuming the array size is n?

- ○ $\mathcal{O}(1)$
- [✓ $\mathcal{O}(n \times \log n)$]
- ○ $\mathcal{O}(n)$
- ○ $\mathcal{O}(n^2)$

Feedback Form

Weekly Workshop Feedback Form

Question 1

I am enrolled in:

- ○ FIT1008
- ○ FIT2085
- ○ FIT1054

Question 2

What needs improvement?

No response

Question 3

What worked best?

No response

Question 4

How engaged were you by the workshop?

- ○ □□□ Very engaged
- ○ □□□ Engaged
- ○ □😐□ Not impressed
- ○ 😕😴□ Lost