Algorithm analysis

Lecture 2 — 21/10/2021

- **Correctness**: Se o algoritmo executa rapidamente e bem, como era esperado.
- **Efficiency**: Desempenho do algoritmo (tempo demorado e espaço ocupado).

	theoretical analysis	*experimental analysis*
correctness	proof or correctness argumentation	pre-defined or random tests*
efficiency (time and space)	complexity	performance tests

O que vamos estudar!

Specifications

- Para provar que um algoritmo executa corretamente:
  - A striet problem specification.
  - A strict algorithm specification.

Preconditions and postconditions

`double squareRoot(double x)`
- → preconditions: `x>=0`
- → postconditions: `RESULT * RESULT = x`
  `RESULT >= 0`

`template <typename T> void sort(vector<T> v)`
- → preconditions: comparision operators defined in T.
- → postconditions: `v[0] <= v[1] <= (...) <= v[n-1]`
  v tem os mesmos elementos do início.

Partial and correctness

- Partial correctness: Caso consiga acabar por causa dos inputs introduzidos.
  Ex: Divisão por 0 dá erro.

- Total correctness: O programa acaba SEMPRE e acaba bem!

Invariant and loop invariant

Para provar que o ciclo está correto:
- Provar que a invariante/expressão é verdadeira no início do ciclo (inicialization)
- Provar que a invariante é verdadeira em todas as iterações, caso se assuma que é verdadeira no início (maintenance)
- Provar que a invariante é verdadeira no fim do ciclo (termination)

Para provar que o ciclo termina:
- Procurar uma variante do ciclo.