

# Amortizovaná složitost – Příklad 1

## Každý třetí

Mějme datovou strukturu zřetěžený seznam s operacemi:

- ▶  $\text{ADD}(x)$  – přidá prvek na konec seznamu
- ▶  $\text{REMOVEEACHTHIRD}$  – projde celý seznam a odstraní každý třetí prvek seznamu

Skutečná cena operací:

- ▶  $\text{ADD}$ : 1
- ▶  $\text{REMOVEEACHTHIRD}$ :  $n$ , kde  $n$  je aktuální délka seznamu

Dokažte, že amortizovaná složitost obou operací je konstantní.

## Amortizovaná složitost – Příklad 2

### Seřazený zásobník

Mějme zásobník, který podporuje následující operace:

- ▶  $\text{ORDEREDPUSH}(x)$  – smaže z vrcholu zásobníku všechny prvky menší než  $x$  a pak na vrchol vloží  $x$
- ▶  $\text{POP}(x)$  – odstraní prvek z vrcholu zásobníku (a vrátí jej)

Tento zásobník implementujeme pomocí jednostranně zřetěženého seznamu. Dokažte, že amortizovaná časová složitost obou operací je konstantní.

## Amortizovaná složitost – Příklad 3

**Binární počítadlo** implementované jako „neomezené“ pole bitů (na počátku 0).

- ▶ Mějme jednu operaci INCREMENT. Dokažte, že je její amortizovaná cena konstantní.

## Amortizovaná složitost – Příklad 3

**Binární počítadlo** implementované jako „neomezené“ pole bitů (na počátku 0).

- ▶ Mějme jednu operaci `INCREMENT`. Dokažte, že je její amortizovaná cena konstantní.
- ▶ Přidejme navíc operaci `RESET`, která počítadlo vynuluje. Rozmyslete, jak tuto operaci implementovat a dokažte, že amortizovaná složitost obou operací je stále konstantní.

## Amortizovaná složitost – Příklad 3

**Binární počítadlo** implementované jako „neomezené“ pole bitů (na počátku 0).

- ▶ Mějme jednu operaci `INCREMENT`. Dokažte, že je její amortizovaná cena konstantní.
- ▶ Přidejme navíc operaci `RESET`, která počítadlo vynuluje. Rozmyslete, jak tuto operaci implementovat a dokažte, že amortizovaná složitost obou operací je stále konstantní.
- ▶ Co kdybychom chtěli mít i operaci `DECREMENT`?