Matéria: Estruturas de dados e análise de algoritmos
Data de entrega: 2024-10-02

Descrição

Para n = 100, n = 100 mil e n = 1 milhão
Use arrays
Compare o tempo gato para os diferentes valores de n
Postar o código em um arquivo PDF, ou link do repositório também em PDF.
Pode enviar mais de um arquivo

Teste com 100 valores:

```
use const_random::const_random;


fn max_number(numbers: &[u64]) -> Option<u64> {
        let mut large = numbers.first()?;

                for number in numbers {
                        if large < number {
                                large = number
                        }
                }

        Some(*large)
}


fn main() {
        static NUMBERS: [u64; 100] = [const_random!(u64); 100];
```

```
        benchmarking::warm_up();


        let benchmark_result = benchmarking::measure_function(|measurer| {
                measurer.measure(|| {
                        max_number(&NUMBERS);
                });
        }).unwrap();


        println!("{:?}", benchmark_result.elapsed());
}
```

Output: `1.442µs`

Teste com 100 mil valores:

```
use const_random::const_random;


fn max_number(numbers: &[u64]) -> Option<u64> {
        let mut large = numbers.first()?;


                for number in numbers {
                        if large < number {
                                large = number
                        }
                }


        Some(*large)
}
```

```
fn main() {
        static NUMBERS: [u64; 100_000] = [const_random!(u64); 100_000];
        benchmarking::warm_up();

        let benchmark_result = benchmarking::measure_function(|measurer| {
                measurer.measure(|| {
                        max_number(&NUMBERS);
                });
        }).unwrap();

        println!("{:?}", benchmark_result.elapsed());
}
```

Output: `1.564205ms`

Teste com 1 milhão de valores:

```
use const_random::const_random;

fn max_number(numbers: &[u64]) -> Option<u64> {
        let mut large = numbers.first()?;

                for number in numbers {
                        if large < number {
                                large = number
                        }
```

```
                }


        Some(*large)

}


fn main() {

        static NUMBERS: [u64; 1_000_000] = [const_random!(u64);
1_000_000];

        benchmarking::warm_up();


        let benchmark_result = benchmarking::measure_function(|measurer| {

                measurer.measure(|| {

                        max_number(&NUMBERS);

                });

        }).unwrap();


        println!("{:?}", benchmark_result.elapsed());

}
```

Output: `13.162994ms`

[Repositório no GitHub]