🦀

計算機科学基礎実験
第一回レポート
Rust によるプログラミング演習

情報科学科
Rai

2023 年 12 月 3 日

1. 課題内容

1.1. 課題 1: Rust でハローワールドを出力する

Rust のプログラミング言語を使用して、コンソールに「Hello, World!」と出力するプログラムを作成してください。

プログラムの実行結果は コード 1 のようになることを確認してください。

コード 1

```
$ cargo run
   Compiling hello-world v0.1.0 (/home/r4ai/Projects/hello-world)
    Finished dev [unoptimized + debuginfo] target(s) in 0.25s
     Running `target/debug/hello-world`
Hello, World!
```

1.2. 課題 2: Rust で FizzBuzz を実装する

Rustのプログラミング言語を使用して、1から100までの整数を順番に出力するプログラムを作成してください。

ただし、以下の条件を満たすようにしてください。

- 3 の倍数の場合は「Fizz」を出力する
- 5 の倍数の場合は「Buzz」を出力する
- 3 の倍数かつ 5 の倍数の場合は「FizzBuzz」を出力する

プログラムの実行結果は以下のようになることを確認してください。

```
$ cargo run
   Compiling fizzbuzz v0.1.0 (/home/r4ai/Projects/fizzbuzz)
    Finished dev [unoptimized + debuginfo] target(s) in 0.25s
     Running `target/debug/fizzbuzz`
1
2
Fizz
4
Buzz
Fizz
7
... (中略)
94
```

```
Buzz
Fizz
97
98
Fizz
Buzz
```

1.3. 課題 3: Rust でクイックソートを実装する

Rust のプログラミング言語を使用して、クイックソートを実装してください。

クイックソートは、与えられたいくつかの要素を、所定の順序にしたがって並び替えるソートアルゴリズムです。 クイックソートには次の特徴があります (要素の個数を N とします)。

- 部分問題を再帰的に解くアルゴリズムである (分割統治法)
- 最悪時の計算量は $O(N^2)$
- 最良時と平均時の計算量は $O(N \log N)$

以下に示すアルゴリズムは、サイズ N の配列 A を昇順に並び替えるクイックソートの動作を表したものです。

$A[X] \left(X = \lfloor \frac{N}{2} \rfloor\right)$ を軸としてソートを行う。

1. 空の配列 L, R を用意し、次の操作を $i = 0, 1, \cdots, N-1$ について行う。
    1. $i = X$ ならば何も行わない。
    2. $i \neq X$ かつ $A[i] < A[X]$ ならば $A[i]$ を L の末尾に追加する。
    3. $i \neq X$ かつ $A[i] \geq A[X]$ ならば $A[i]$ を R の末尾に追加する。
2. L, R をクイックソートを用いて再帰的にソートする。空配列の場合は何も変わらない。
3. L の要素、 $A[X], R$ の要素をこの順につなげてできる配列を出力する。

参考文献：アルゴ式「Q.4 クイックソート」、https://algo-method.com/tasks/442

プログラムの実行結果は以下のようになることを確認してください。

```
$ cargo run
   Compiling quicksort v0.1.0 (/home/r4ai/Projects/quicksort)
    Finished dev [unoptimized + debuginfo] target(s) in 0.25s
     Running `target/debug/quicksort`
1
2
3
4
5
```

```
6
7
8
9
10
```

2. アルゴリズム

2.1. 課題 1: Rust でハローワールドを出力する

以下に、ハローワールドを出力するアルゴリズムを示す。

1. 「Hello, World!」を出力する。

2.2. 課題 2: Rust で FizzBuzz を実装する

以下に、FizzBuzz を実装するアルゴリズムを示す。

1. $i = 1, 2, \cdots, 100$ について、次の操作を行う。
    1. i が 3 の倍数かつ 5 の倍数ならば「FizzBuzz」を出力する。
    2. i が 3 の倍数ならば「Fizz」を出力する。
    3. i が 5 の倍数ならば「Buzz」を出力する。
    4. いずれでもない場合は i を出力する。

2.3. 課題 3: Rust でクイックソートを実装する

以下に、クイックソートを実装するアルゴリズムを示す。

1. $A[X]\left(X = \left\lfloor \frac{N}{2} \right\rfloor\right)$ を軸としてソートを行う。
2. 空の配列 L, R を用意し、次の操作を $i = 0, 1, \cdots, N - 1$ について行う。
    1. $i = X$ ならば何も行わない。
    2. $i \neq X$ かつ $A[i] < A[X]$ ならば $A[i]$ を L の末尾に追加する。
    3. $i \neq X$ かつ $A[i] \geq A[X]$ ならば $A[i]$ を R の末尾に追加する。
3. L, R をクイックソートを用いて再帰的にソートする。空配列の場合は何も変わらない。
4. L の要素、 $A[X], R$ の要素をこの順につなげてできる配列を出力する。

3. プログラム

3.1. 課題 1: Rust でハローワールドを出力する

節 2.1 を Rust で実装したプログラムを コード 2 に示す。

コード 2: helloworld.rs

```rust
fn main() {
    println!("Hello, World!");
}
```

3.2. 課題 2: Rust で FizzBuzz を実装する

節 2.2 を Rust で実装したプログラムを コード 3 に示す。

コード 3: fizzbuzz.rs

```rust
fn main() {
    for i in 1..=100 {
        if i % 15 == 0 {
            println!("FizzBuzz");
        } else if i % 3 == 0 {
            println!("Fizz");
        } else if i % 5 == 0 {
            println!("Buzz");
        } else {
            println!("{}", i);
        }
    }
}
```

3.3. 課題 3: Rust でクイックソートを実装する

節 2.3 を Rust で実装したプログラムを コード 4 に示す。

コード 4: quicksort.rs

```rust
fn quicksort<T: Ord + Copy>(a: &mut [T]) {
    if a.len() <= 1 {
        return;
    }
```

```rust

    let pivot = a.len() / 2;
    let mut l = Vec::new();
    let mut r = Vec::new();

    for i in 0..a.len() {
        if i == pivot {
            continue;
        }

        if a[i] < a[pivot] {
            l.push(a[i]);
        } else {
            r.push(a[i]);
        }
    }

    quicksort(&mut l);
    quicksort(&mut r);

    for i in 0..l.len() {
        a[i] = l[i];
    }
    a[l.len()] = a[pivot];
    for i in 0..r.len() {
        a[l.len() + 1 + i] = r[i];
    }
}

fn main() {
    let mut a = [6, 3, 8, 2, 9, 1, 4, 7, 5, 10];
    quicksort(&mut a);
    for i in 0..a.len() {
        println!("{}", a[i]);
    }
}
```

4. 実行結果

4.1. 課題 1: Rust でハローワールドを出力する

コード 2 を実行した結果を 図 1 に示す。

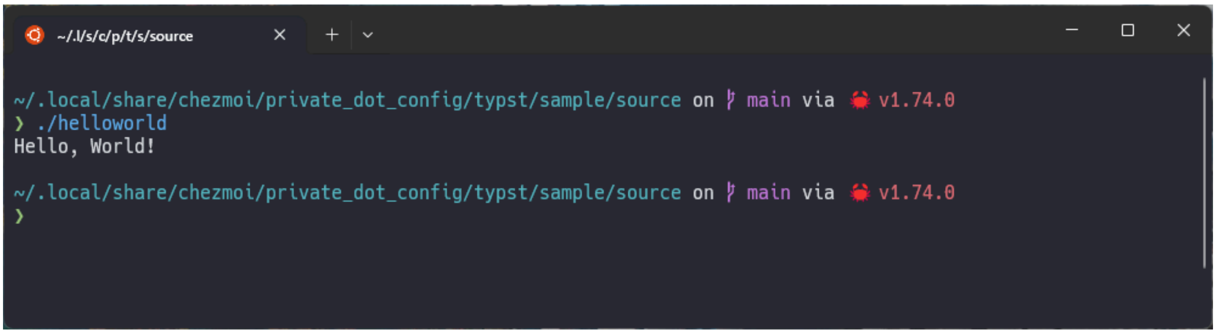

図 1: helloworld.rs の実行結果

4.2. 課題 2: Rust で FizzBuzz を実装する

コード 3 を実行した結果を 図 2 に示す。なお、出力結果が長すぎるため、一部のみを抜粋している。

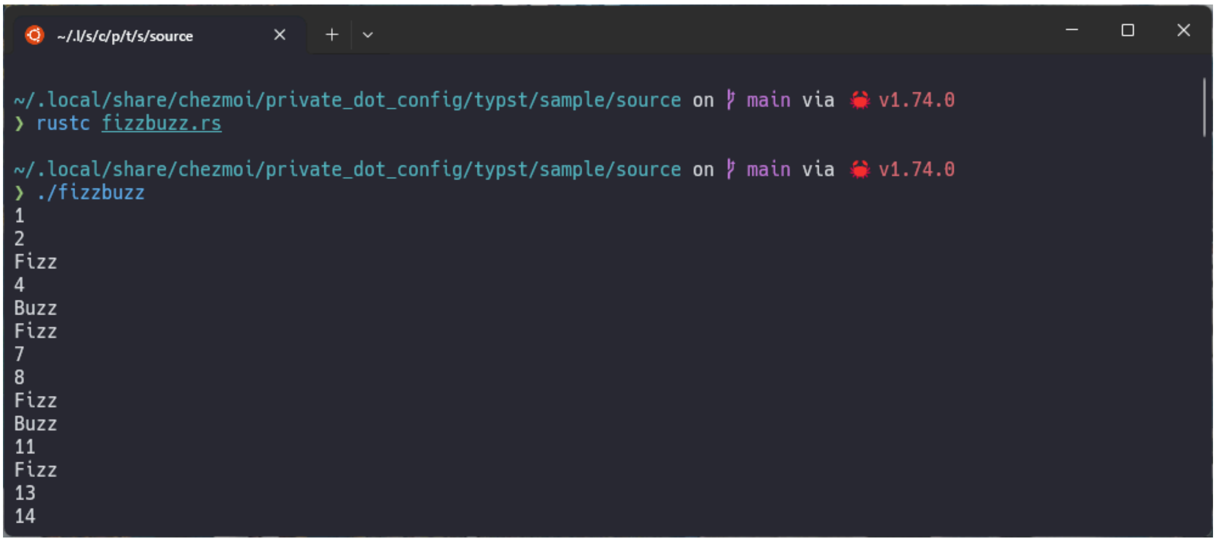

図 2: fizzbuzz.rs の実行結果

4.3. 課題 3: Rust でクイックソートを実装する

コード 4 を実行した結果を 図 3 に示す。

図 3: quicksort.rs の実行結果

5. 考察

5.1. 課題 1: Rust でハローワールドを出力する

Rust のプログラミング言語を使用して、コンソールに「Hello, World!」と出力するプログラムを作成した。

5.2. 課題 2: Rust で FizzBuzz を実装する

Rust のプログラミング言語を使用して、1 から 100 までの整数を順番に出力するプログラムを作成した。

5.3. 課題 3: Rust でクイックソートを実装する

Rust のプログラミング言語を使用して、クイックソートを実装した。クイックソートの計算量は、最悪時は 式 (1) 、最良時と平均時は 式 (2) である。

$$O(N^2) \tag{1}$$

$$O(N \log N) \tag{2}$$

よって、クイックソートはバブルソートや挿入ソートと比較して、計算量が少ないアルゴリズムであると言える。また、クイックソートは並列化が容易であるため、並列計算に適しており、さらなる高速化が期待できる。

6. おまけ

6.1. 有名分布

6.1.1. 正規分布

X を確率変数、 $\mu \in \mathbb{R}, \sigma > 0$ とする。 X の確率密度関数が、

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]$$

となるとき、 X は平均 μ 、分散 σ^2 の正規分布に従うといい、 $X \sim N(\mu, \sigma^2)$ と表す。

6.1.1.1. 積率母関数の導出

$$\begin{aligned}
E[e^{tX}] &= \int_{-\infty}^{\infty} e^{tx} f(x) \ \mathrm{d}x \\
&= \int_{-\infty}^{\infty} e^{tx} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right] \mathrm{d}x \\
&= \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{\infty} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2} + tx\right] \mathrm{d}x \\
&= \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{\infty} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2} + tx\right] \mathrm{d}x \\
&= \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{\infty} \exp\left[-\frac{\{x-(\mu+\sigma^2 t)\}^2}{2\sigma^2} + \mu t + \frac{\sigma^2 t^2}{2}\right] \mathrm{d}x \\
&= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[\mu t + \frac{\sigma^2 t^2}{2}\right] \int_{-\infty}^{\infty} \exp\left[-\frac{\{x-(\mu+\sigma^2 t)\}^2}{2\sigma^2}\right] \mathrm{d}x
\end{aligned}$$

ここで、 $y = \dfrac{x-(\mu+\sigma^2 t)}{\sqrt{2\sigma^2}}$ とおくと、 $\mathrm{d}y = \dfrac{1}{\sqrt{2\sigma^2}} \mathrm{d}x$ であり、

$$\begin{aligned}
E[e^{tX}] &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[\mu t + \frac{\sigma^2 t^2}{2}\right] \sqrt{2\sigma^2} \int_{-\infty}^{\infty} e^{-y^2} \, \mathrm{d}y \\
&= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[\mu t + \frac{\sigma^2 t^2}{2}\right] \sqrt{2\pi\sigma^2} \qquad \left(\because \int_{-\infty}^{\infty} e^{-x^2} \, \mathrm{d}x = \sqrt{\pi}\right) \\
&= \exp\left[\mu t + \frac{\sigma^2 t^2}{2}\right]
\end{aligned}$$

を得る。

よって、正規分布の積率母関数は、

$$E[e^{tX}] = \exp\left[\mu t + \frac{\sigma^2 t^2}{2}\right]$$

である。

6.1.1.2. 期待値と分散の導出

積率母関数より、正規分布の期待値は、

$$
\begin{aligned}
E[X] &= \frac{\mathrm{d}}{\mathrm{d}t} E[e^{tX}] \Big|_{t=0} \\
&= \frac{\mathrm{d}}{\mathrm{d}t} \exp\left[\mu t + \frac{\sigma^2 t^2}{2}\right] \Bigg|_{t=0} \\
&= (\mu + \sigma^2 t) \exp\left[\mu t + \frac{\sigma^2 t^2}{2}\right] \Bigg|_{t=0} \\
&= \mu
\end{aligned}
\tag{3}
$$

である。

また、正規分布の 2 次積率は、

$$
\begin{aligned}
E[X^2] &= \frac{\mathrm{d}^2}{\mathrm{d}t^2} E[e^{tX}] \Big|_{t=0} \\
&= \frac{\mathrm{d}}{\mathrm{d}t} (\mu + \sigma^2 t) \exp\left[\mu t + \frac{\sigma^2 t^2}{2}\right] \Bigg|_{t=0} \\
&= \sigma^2 \exp\left[\mu t + \frac{\sigma^2 t^2}{2}\right] + (\mu + \sigma^2 t)^2 \exp\left[\mu t + \frac{\sigma^2 t^2}{2}\right] \Bigg|_{t=0} \\
&= \sigma^2 + \mu^2
\end{aligned}
\tag{4}
$$

である。よって、 式 (3) と 式 (4) より正規分布の分散は、

$$
\begin{aligned}
V[X] &= E[X^2] - E[X^2] \\
&= \sigma^2 + \mu^2 - \mu^2 \\
&= \sigma^2
\end{aligned}
$$

である。