소프트웨어 마에스트로 특강

Rust로 알고리즘 문제 풀어보기

Chris Ohk
utilForever@gmail.com

발표자 소개

소프트웨어 마에스트로 특강
Rust로 알고리즘 문제 풀어보기

- 옥찬호 (Chris Ohk)
  - (현) Momenti Engine Engineer
  - (전) Nexon Korea Game Programmer
  - Microsoft Developer Technologies MVP
  - C++ Korea Founder & Administrator
  - Reinforcement Learning KR Administrator
  - IT 전문서 집필 및 번역 다수
    - 게임샐러드로 코드 한 줄 없이 게임 만들기 (2013)
    - 유니티 Shader와 Effect 제작 (2014)
    - 2D 게임 프로그래밍 (2014), 러스트 핵심 노트 (2017)
    - 모던 C++ 입문 (2017), C++ 최적화 (2019)

utilForever@gmail.com
utilForever

목차

PS에서의 Rust

- PS에서 주로 사용하는 언어(C++, Python 등)에 비해 어려운 점이 많다.
  - 불편한 콘솔 입출력
    - C++ : `std::cin, std::cout`
    - Python : `input.split(), print()`
    - Rust : 간단하지 않음 (1줄로 안됨)
  - 엄격한 언어 문법
    - 변수 타입을 반드시 일치시켜야 한다.
    - 수명을 고려하면서 소유권 규칙을 지켜야 한다.
  - 부실한 자료 구조와 알고리즘
    - 가장 기본적인 자료 구조와 알고리즘만 존재한다.
    - C++이나 Python에 있는 다양한 함수들이 Rust에는 존재하지 않는다.

PS에서의 Rust

- 그러면 굳이 왜?
  - 불편한 콘솔 입출력
    - 아 이건 답이 없... (X를 눌러 조의를 표하세요.)
    - 특히 입력 라인 수가 많을수록 매우 불리해진다.
  - 엄격한 언어 문법
    - 품질이 높은 코드를 작성하게 된다.
    - 메모리 관리를 잘 하고 있는지 확인할 수 있다.
  - 부실한 자료 구조와 알고리즘
    - 자료 구조와 알고리즘을 직접 구현해보며 이해하는 게 도움이 된다.

A + B (BOJ 1000) - C++


```cpp
#include <iostream>

int main() {
    int a, b;

    std::cin >> a >> b;
    std::cout << a + b << std::endl;

    return 0;
}
```

A + B (BOJ 1000) - Rust


```rust
use std::io;

fn main() {
    let mut s = String::new();

    io::stdin().read_line(&mut s).unwrap();

    let values:Vec<i32> = s
        .as_mut_str()
        .split_whitespace()
        .map(|s| s.parse().unwrap())
        .collect();

    println!("{}", values[0] + values[1]);
}
```


ㅋㅋㅋㅋㅋㅋ
정신이 들어?
역시 타임코스모스야..
성능 확실하구만.

Fast I/O in Rust

소프트웨어 마에스트로 특강
Rust로 알고리즘 문제 풀어보기

```rust
pub struct UnsafeScanner<R> {
    reader: R,
    buf_str: Vec<u8>,
    buf_iter: str::SplitAsciiWhitespace<'static>,
}

impl<R: io::BufRead> UnsafeScanner<R> {
    pub fn new(reader: R) -> Self {
        Self {
            reader,
            buf_str: vec![],
            buf_iter: "".split_ascii_whitespace(),
        }
    }

    pub fn token<T: str::FromStr>(&mut self) -> T {
        loop {
            if let Some(token) = self.buf_iter.next() {
                return token.parse().ok().expect("Failed parse");
            }
            self.buf_str.clear();
            self.reader
                .read_until(b'\n', &mut self.buf_str)
                .expect("Failed read");
            self.buf_iter = unsafe {
                let slice = str::from_utf8_unchecked(&self.buf_str);
                std::mem::transmute(slice.split_ascii_whitespace())
            }
        }
    }
}
```

Fast I/O in Rust

소프트웨어 마에스트로 특강
Rust로 알고리즘 문제 풀어보기

```rust
fn main() {
    let (stdin, stdout) = (io::stdin(), io::stdout());
    let mut scan = UnsafeScanner::new(stdin.lock());
    let mut out = io::BufWriter::new(stdout.lock());

    let (a, b) = (scan.token::<i64>(), scan.token::<i64>());

    writeln!(out, "{}", a + b).unwrap();
}
```

GCD와 LCM (BOJ 2609) - C++


```cpp
#include <iostream>
#include <numeric>

int main(int argc, char* argv[])
{
    int a, b;
    std::cin >> a >> b;

    std::cout << std::gcd(a, b) << '\n' << std::lcm(a, b) << '\n';

    return 0;
}
```

GCD와 LCM (BOJ 2609) - Rust

std::gcd, std::lcm?

그런 거 없다.

GCD와 LCM (BOJ 2609) - Rust

소프트웨어 마에스트로 특강
Rust로 알고리즘 문제 풀어보기

```rust
fn lcm(first: i32, second: i32) -> i32 {
    first * second / gcd(first, second)
}

fn gcd(first: i32, second: i32) -> i32 {
    let mut max = first;
    let mut min = second;

    if min > max {
        let val = max;

        max = min;
        min = val;
    }

    loop {
        let res = max % min;

        if res == 0 {
            return min;
        }

        max = min;
        min = res;
    }
}
```

다음 순열 (BOJ 10972) - C++


```cpp
int main(int argc, char* argv[])
{
    int n;
    std::cin >> n;

    std::vector<int> nums;
    nums.reserve(n);

    for (int i = 0; i < n; ++i) {
        int num;
        std::cin >> num;

        nums.emplace_back(num);
    }

    bool res = std::next_permutation(nums.begin(), nums.end());

    if (res) {
        for (auto& num : nums) {
            std::cout << num << ' ';
        }
    }
    else {
        std::cout << "-1";
    }

    std::cout << '\n';
}
```

다음 순열 (BOJ 10972) - Rust

std::next_permutation?

어림도 없다.

다음 순열 (BOJ 10972) - Rust


```rust
pub fn next_permutation(nums: &mut Vec<i32>) -> bool {
    let last_ascending = match nums.windows(2).rposition(|w| w[0] < w[1]) {
        Some(i) => i,
        None => {
            nums.reverse();
            return false;
        }
    };

    let swap_with = nums[last_ascending + 1..]
        .binary_search_by(|n| i32::cmp(&nums[last_ascending], n).then(Ordering::Less))
        .unwrap_err();
    nums.swap(last_ascending, last_ascending + swap_with);
    nums[last_ascending + 1..].reverse();

    true
}
```

Rust 자료 구조

- 기본 자료 구조들은 구현되어 있다. 다만, 익숙한 이름이 아닐 뿐...
  - 스택(Stack) : ~~std::stack~~ **std::collections::Vec**
  - 큐(Queue) / 덱(Deque) : ~~std::queue, std::deque~~ **std::collections::VecDeque**
  - 링크드 리스트(Linked List) : ~~std::list~~ **std::collections::LinkedList**
  - 맵(Map) : ~~std::map~~ **std::collections::HashMap**
  - B-트리(B-Tree) : **std::collections::BTreeMap**
  - 셋(Set) : ~~std::set~~ **std::collections::HashSet/BTreeSet**
  - 우선순위 큐(Priority Queue) : ~~std::priority_queue~~ **std::collections::BinaryHeap**

계단 오르기 (BOJ 2579) - Rust


```rust
fn main() {
    let num_stairs = input_integers()[0];

    let mut stairs = vec![0; num_stairs as usize + 1];

    for i in 1..=num_stairs as usize {
        stairs[i] = input_integers()[0];
    }

    let mut scores = vec![vec![0; 3]; num_stairs as usize + 1];

    scores[1][1] = stairs[1];

    for i in 2..=num_stairs as usize {
        scores[i][1] = cmp::max(scores[i - 2][1], scores[i - 2][2]) + stairs[i];
        scores[i][2] = scores[i - 1][1] + stairs[i];
    }

    println!(
        "{}",
        cmp::max(
            scores[num_stairs as usize][1],
            scores[num_stairs as usize][2]
        )
    );
}
```

제로 (BOJ 10773) - Rust


```rust
use std::{collections::VecDeque, io};

fn main() {
    let n = input_integers()[0];

    let mut nums = VecDeque::new();

    for _ in 0..n {
        let num = input_integers()[0];

        if num == 0 {
            nums.pop_back();
        } else {
            nums.push_back(num);
        }
    }

    let mut sum = 0;

    while !nums.is_empty() {
        sum += nums.back().unwrap();
        nums.pop_back();
    }

    println!("{}", sum);
}
```

BOJ 수열 1 (BOJ 13323) - Rust


```rust
fn main() {
    let n = input_integers()[0];

    let mut queue = BinaryHeap::new();
    let mut ans: i64 = 0;

    let nums = input_integers();

    for i in 0..n {
        let mut num = nums[i as usize];
        num -= i;

        queue.push(num);

        if !queue.is_empty() && queue.peek().unwrap() > &num {
            ans += (queue.peek().unwrap() - num) as i64;
            queue.pop();
            queue.push(num);
        }
    }

    println!("{}", ans);
}
```

알아두면 좋을 몇 가지 팁

- Rust에서 char 타입은 4바이트다.
  트라이(Trie)를 사용하는 문제 중 일부에서 메모리 초과가 발생한다.
  예 - 전설 (BOJ 19585)
- 데이터를 입력받는데 약 10MB의 메모리가 필요하기 때문에
  메모리 제한이 매우 작은 경우 다른 언어를 사용해야 한다.
  예 – LCS 5, 6, 7 (BOJ 18438, 18439, 18440)
- 매우 큰 수를 다뤄야 하는 경우 다른 언어를 사용하는게 좋다.
  예 – Euclid’s Algorithm (BOJ 18496)

Rust 알고리즘 문제 관련 저장소

- Contest Algorithms in Rust (https://github.com/EbTech/rust-algorithms)
- Common Data Structures and Algorithms in Rust (https://github.com/utilForever/algorithm-rs)

Thank you!