让人抓狂的 Rust

作者：tiangao
原文链接：https://ld246.com/article/1587384525043
来源网站：链滴

刚看 Rust 没几天，想着去 leetcode 上练练，找到这么一个题

80. 删除排序数组中的重复项 II

答案很简单，Java 代码答案如下

```
class Solution {
    public int removeDuplicates(int[] nums) {
        int i = 0;
        for (int n : nums)
            if (i < 2 || n > nums[i-2])
                nums[i++] = n;
        return i;
    }
}
```

很简单吧，也看了几天 Rust 语法了，直接上

```
fn remove_duplicates(nums: µ Vec<i32>) -> i32 {
    let mut i = 0;
    for x in nums {
        if i < 2 || x > nums[i-2] {
            nums[i] = x;
        }
        i++;
    }
    return i;
}
```

这也太简单了，然而。。。

```
$ cargo run
101 
   Compiling demo v0.1.0 (/Users/tiangaoshi/rust/demo)
error: expected expression, found `+`
  --> src/main.rs:14:11
   |
14 |         i++;
   |           ^ expected expression

error[E0308]: mismatched types
  --> src/main.rs:11:25
   |
11 |         if i < 2 || x > nums[i-2] {
   |                      ^^^^^^^^^^
   |                      |
   |                      expected `µ i32`, found `i32`
   |                      help: consider mutably borrowing here: `µ nums[i-2]`

error[E0308]: mismatched types
  --> src/main.rs:12:23
```

```
  |
12 |             nums[i] = x;
  |                     ^
  |                     |
  |                     expected `i32`, found `µ i32`
  |                     help: consider dereferencing the borrow: `*x`

error[E0308]: mismatched types
 --> src/main.rs:16:12
  |
8 | fn remove_duplicates(nums: µ Vec<i32>) -> i32 {
  |                                      --- expected `i32` because of return type
...
16 |     return i;
  |            ^ expected `i32`, found `usize`
  |
help: you can convert an `usize` to `i32` and panic if the converted value wouldn't fit
  |
16 |     return i.try_into().unwrap();
  |            ^^^^^^^^^^^^^^^^^^^^^^^^
```

WTF??? 这么多错误， let mut 不是变量吗，nums 不是数组吗，为啥?

没办法，碰到问题解决问题，

第一个最简单，不支持 `++`, 那就 `i += 1;`

第二个没看懂啥意思，大概是不可变，加一行 `let a = i - 2;`

第三个有提示了 `nums[i] = *x;`

最后一个应该是类型错误

```
fn remove_duplicates(nums: µ Vec<i32>) -> i32 {
    let mut i = 0;
    for x in nums {
        let a = i - 2;
        if i < 2 || *x > nums[a] {
            nums[i] = *x;
        }
        i += 1;
    }
    return i as i32;
}
```

gogogo~~~~

```
└─$ cargo run
    101 
   Compiling demo v0.1.0 (/Users/tiangaoshi/rust/demo)
error[E0382]: borrow of moved value: `nums`
 --> src/main.rs:12:26
  |
8 | fn remove_duplicates(nums: µ Vec<i32>) -> i32 {
```

```
  |                    ---- move occurs because `nums` has type `µ std::vec::Vec<i32>`, which does
ot implement the `Copy` trait
9 |     let mut i = 0;
10 |     for x in nums {
  |             ----
  |             |
  |             value moved here
  |             help: consider borrowing to avoid moving into the for loop: `#`
11 |         let a = i - 2;
12 |         if i < 2 || *x > nums[a] {
  |                      ^^^^ value borrowed here after move

error: aborting due to previous error
```

我真是黑人问号脸了，什么鬼啊

此处省去几天时间的各种尝试和查阅文档，

甚至仔仔细细读了 <<Rust编程之道>> 前面几章，

中间还找到了一个仓库 https://github.com/aylei/leetcode-rust ，直接去翻 80

https://github.com/aylei/leetcode-rust/blob/master/src/solution/s0080_remove_duplicates_from_sorted_array_ii.rs

oh shit~

今天综合几天的经验，终于尝试搞出了一个能过的版本

```
fn remove_duplicates(nums: µ Vec<i32>) -> i32 {
    let mut i = 0;
    let list = nums.clone();
    for x in list {
        if i < 2 {
            let a = i;
            nums[a] = x;
            i += 1;
        } else {
            let a = i;
            let b = i - 2;
            if x > nums[b] {
                nums[a] = x;
                i += 1;
            }
        }
    }
    return i as i32;
}
```

我真的是醉了，这是个什么鬼语言，好歹我也是写过 C 玩过汇编单片机，C++入过门，N 年 Java，年 Golang 滚过， Kotlin 爽过，中间还掺杂 Elixir Swift 认知的职业搬砖工，这写第一个算法题就

我这么久，坑啊

只能说自己菜了，果然老人言说得对，一上来就写代码肯定会栽跟头的

Rust 肯定是值得学习的，看看人家 actix，简直是一骑绝尘

https://www.techempower.com/benchmarks/#section=data-r18&hw=ph&test=update

学无止境，目测最最后一版也只是能过而已，肯定有更精炼的写法，手动捂脸，

最后附上一个此题的提交记录，同一套算法

提交时间	提交结果	执行用时	内存消耗	语言
几秒前	通过	1 ms	39.8 MB	Java
1 小时前	通过	0 ms	2.1 MB	Rust
7 天前	通过	4 ms	3 MB	Golang

第二天

感觉好像又理解了些东西，又写了一版，好看多了，舒服

```
impl Solution {
    fn remove_duplicates(nums: μ Vec<i32>) -> i32 {
        let mut ans = 2;
        let size = nums.len();
        for i in 2..size {
            let a = ans - 2;
            if nums[a] < nums[i] {
                nums[ans] = nums[i];
                ans += 1;
            }
        }
        return ans as i32;
    }
}
```

觉得学语言其实是在学模式，解决一个问题有千万种方法，不通的方法对应的也是不通的效率，开发率跟运行效率就像鱼跟熊掌，enmmm，小孩子才做选择，我都要