拉勾教育
—互联网人实战大学—

数组：实现整数的数字反转

简单/数组、数学、遍历、逆序、原地替换

学习目标

拉勾教育
—互联网人实战大学—

- 了解算法题的解题思路
- 数组的特点
- 数学思维的应用

题目描述

给出一个 32 位的有符号整数，将这个整数每位上的数字进行反转

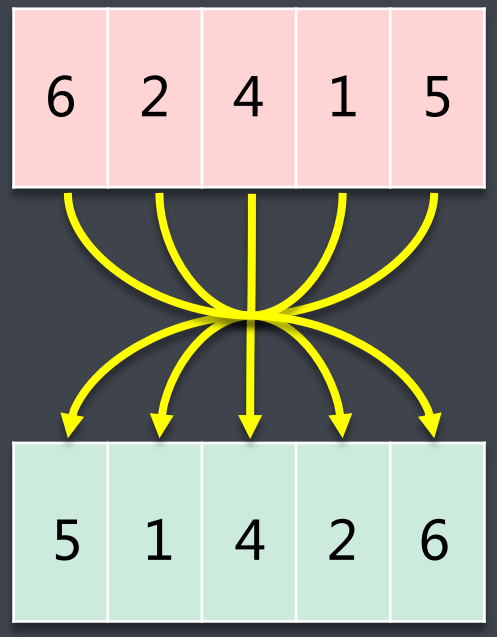
输入：123
输出：321

输入：-123
输出：-321 // 符号不变

输入：120
输出：21 // 首位非0；数值溢出返回0；

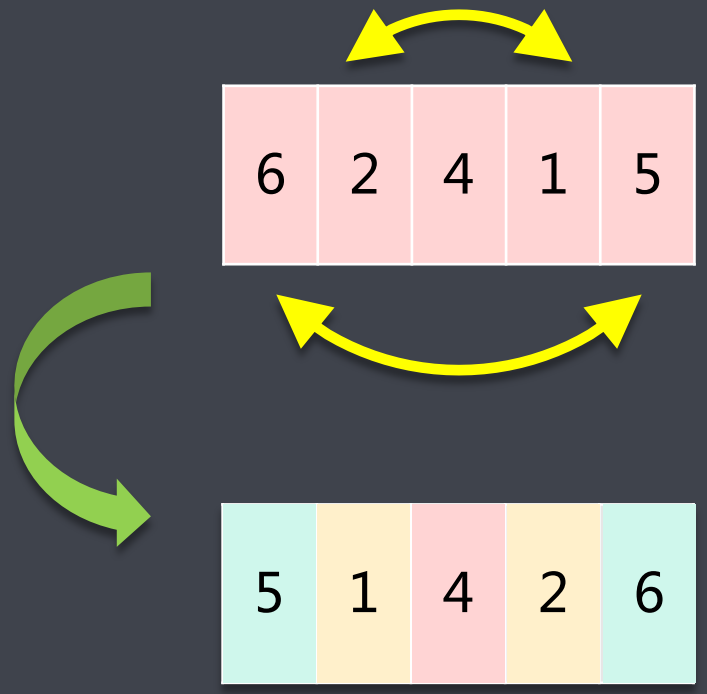
一. Comprehend 理解题意

该题可以理解为一个逆序输出问题



一. Comprehend 理解题意

也是一个首尾交换问题，两端分别对应位置的数字进行交换



二. Choose 数据结构及算法思维选择

拉勾教育
—互联网人实战大学—

方案一：逆序输出（暴力解法）

- 整数转字符串，字符串转数组
- 数据结构：字符数组
- 算法思维：遍历

方案二：首尾交换（优化解法）

- 整数转字符串，字符串转数组
- 数组结构：字符数组
- 算法思维：遍历

二. Choose 数据结构及算法思维选择

拉勾教育
—互联网人实战大学—

数据结构选择

- 字符数组

算法思维选择

- 遍历

索引	元素	物理地址
0	6	0xb000
1	2	0xb020
2	4	0xb040
3	1	0xb060
4	5	0xb080

数组容量固定不变，需在创建数组时指定

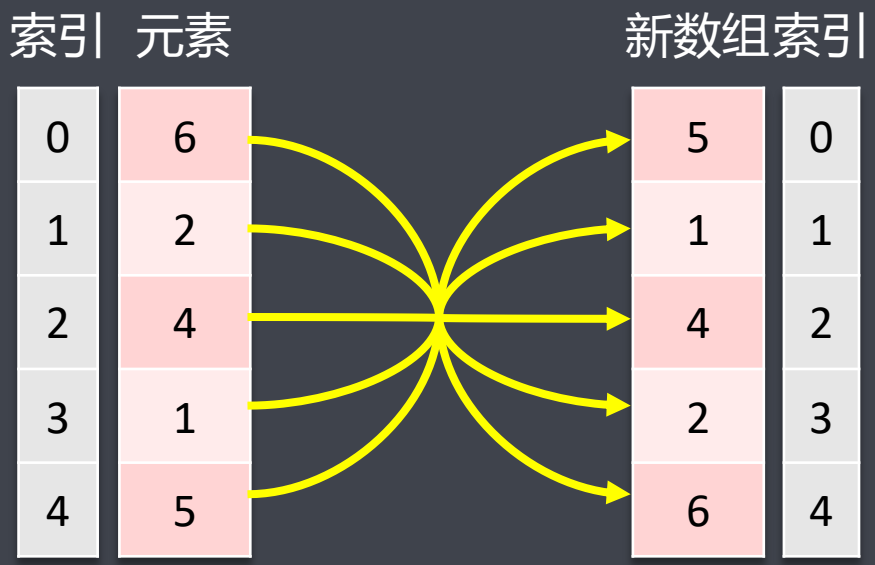
使用连续的物理空间在存取数据

可以通过下标在O(1)的时间复杂度下读取数据

三. Code 基本解法及编码实现

解法一：暴力解法

1. 整数转字符串，再转字符数组
2. 反向遍历字符数组，并将元素存储到新数组中
3. 将新数组转成字符串，再转成整数输出



三. Code 基本解法及编码实现

解法一：暴力解法边界和细节问题

边界问题

- 数组索引越界
- 数值溢出边界：溢出则返回0

细节问题

- 首位不为0
- 符号处理

三. Code 基本解法及编码实现


```java
public int reverse(int x) {
    if(x == Integer.MAX_VALUE) {
        // 整数类型最小值的绝对值 比 最大值的绝对值 大1
        return 0; // 反转必然溢出，返回0
    }
    int sign = x > 0 ? 1 : -1; // 符号
    x = x < 0 ? x * -1 : x; // 无论正负，都当成正数
    // 1.整数转字符串，再转字符数组
    String str = String.valueOf(x);
    char[] chars = str.toCharArray();
    // 2.反向遍历字符数组，并将元素存储到新数组中
    int len = chars.length;
    char[] array = new char[len];
    for (int i = 0; i < len; i++) { // 遍历新数组
        array[i] = chars[len - 1 - i];
    }
    // 3.将新数组转成字符串，再转成整数输出
    long value = Long.valueOf(String.valueOf(array));
    boolean b = value > Integer.MAX_VALUE || value < Integer.MIN_VALUE;
    int result = b ? 0 : (int)value; // 数值越界：溢出则返回0
    return result * sign; // 符号还原
}
```

时间复杂度：O(n)

- 将整数转成字符串O(n)
- 遍历字符数组O(n)
- O(n) +O(n)=O(2n)
- 忽略常数后：O(n)

空间复杂度：O(n)

- 一个字符串O(n)
- 一个新数组O(n)
- O(n) +O(n)=O(2n)
- 忽略常数后：O(n)

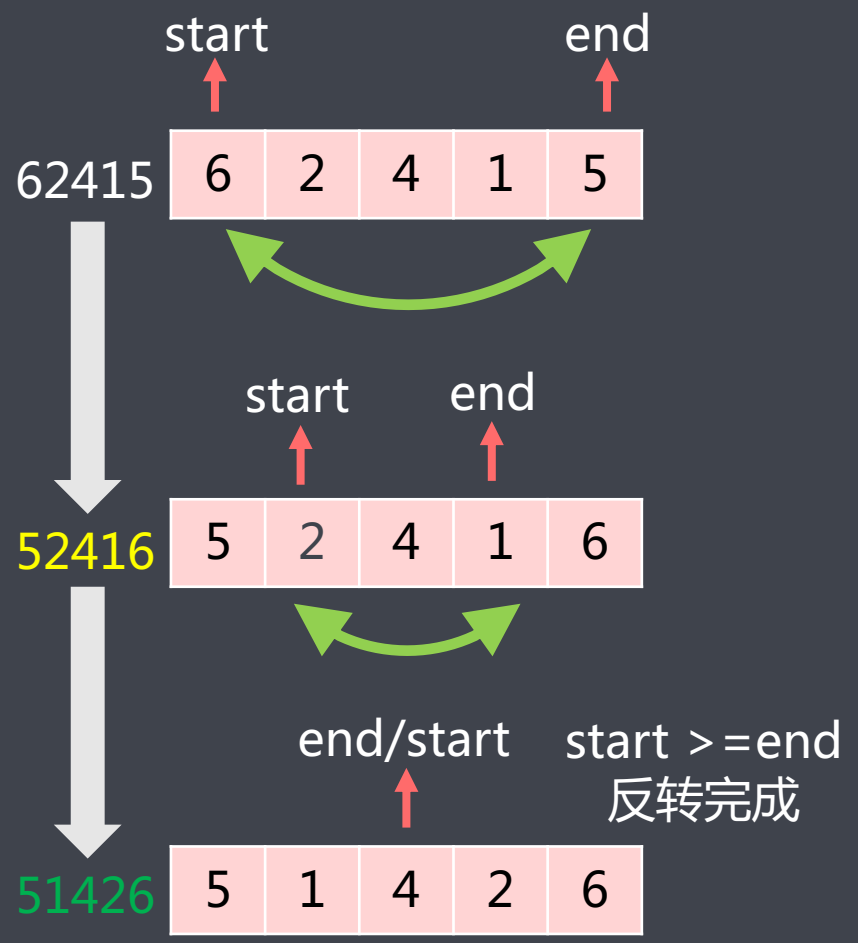
执行耗时:4 ms,击败了15.82% 的Java用户
内存消耗:36.1 MB,击败了60.39% 的Java用户

三. Code 基本解法及编码实现

拉勾教育
—互联网人实战大学—

解法二：数组首尾交换（优化解法）

1. 整数转字符串，再转字符数组
2. 交换首位(start)和末位(end)数字
3. 循环操作：依次交换第二(start++)和倒数第二个(end--)，直到数组剩下1个或0个元素。
4. 将原数组转成字符串，再转成整数输出



三. Code 基本解法及编码实现

解法二：优化解法边界和细节问题

边界问题

- 数组索引越界：数组长度为偶数，反转完成标志为start>end；为奇数时反转完成标志为start==end
- 数值溢出边界：溢出则返回0

细节问题

- 首位不为0
- 符号处理

三. Code 基本解法及编码实现

拉勾教育
—互联网人实战大学—

```java
public int reverse(int x) {
    if(x == Integer.MAX_VALUE) {
        // 整数类型最小值的绝对值 比 最大值的绝对值 大1
        return 0; // 反转必然溢出，返回0
    }
    int sign = x > 0 ? 1 : -1; // 符号
    x = x < 0 ? x * -1 : x; // 无论正负，都当成正数
    // 1.整数转字符串，再转字符数组
    String str = String.valueOf(x);
    char[] chars = str.toCharArray();
    // 2.交换首位(start)和末位(end)数字
    // 3.循环操作：依次交换第二(start++)和倒数第二个(end--)
    int start = 0 , end = chars.length - 1;
    while (start < end) { // 反转完成的标志：start >= end
        // 交换两端等距离的元素
        char temp = chars[start];
        chars[start] = chars[end];
        chars[end] = temp;
        start++;
        end--;
    }
    // 4.将原数组转成字符串，再转成整数输出
    long value = Long.valueOf(String.valueOf(chars));
    boolean b = value > Integer.MAX_VALUE || value < Integer.MIN_VALUE;
    int result = b ? 0 : (int)value;
    return result * sign;
}
```

时间复杂度：O(n)

- 将整数转成字符串O(n)
- 遍历字符数组O(n)
- O(n) +O(n)=O(2n)
- 忽略常数后：O(n)

空间复杂度：O(n)

- 一个字符串O(n)
- ~~一个新数组O(n)~~
- 绝对空间消耗降低：O(n)

执行耗时:3 ms,击败了23.28% 的Java用户
内存消耗:35.6 MB,击败了99.73% 的Java用户

四. Consider 思考更优解

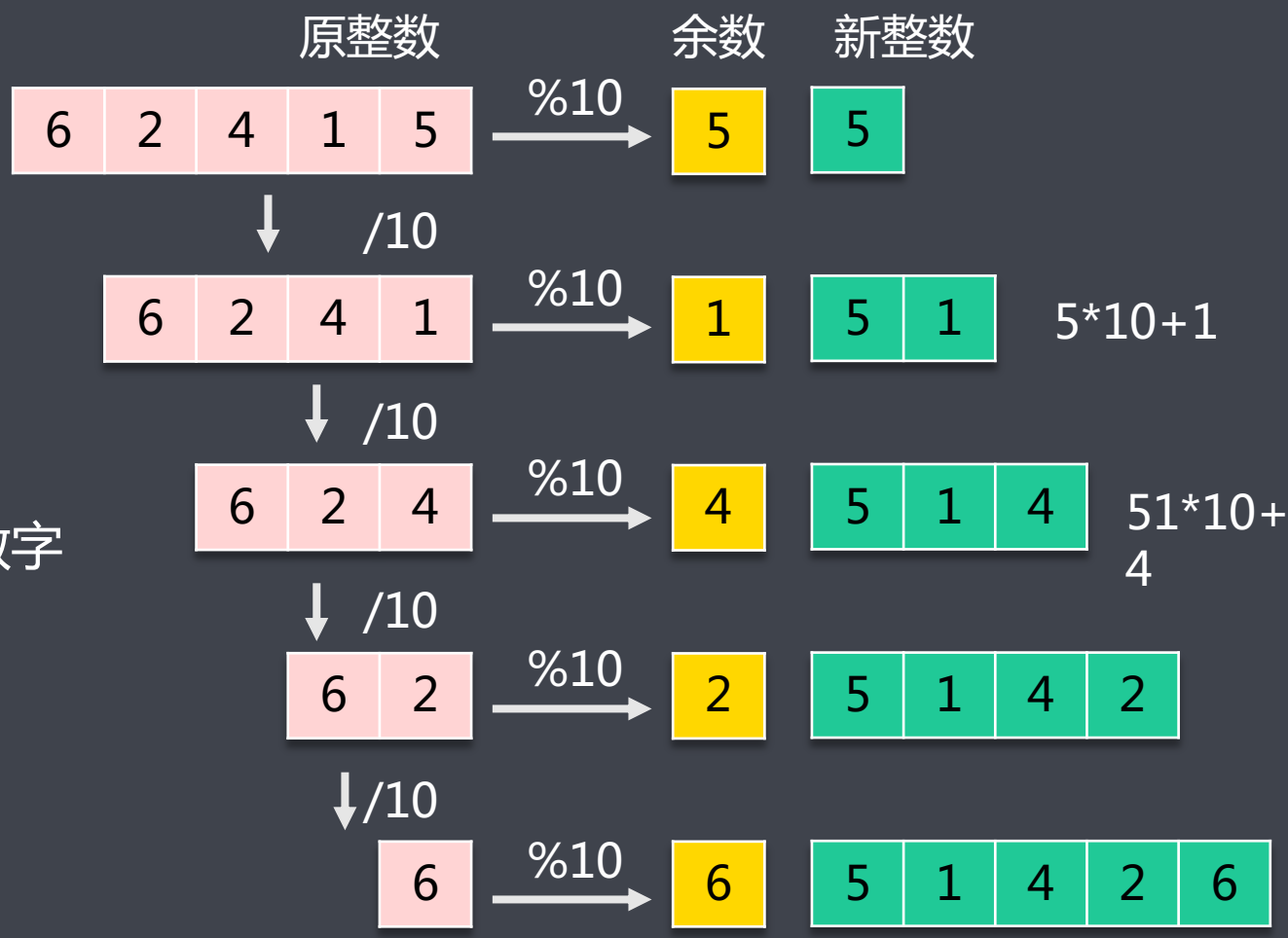
1. 剔除无效代码或优化空间消耗
   - 操作是必须的吗？
   - 数据结构是必须的吗？
2. 寻找更好的算法思维
   - 既然是整数，能否用数学思维？
   - 借鉴其它算法

五. Code 最优解思路及编码实现

拉勾教育
—互联网人实战大学—

最优解：数学思维解法

1. 尝试拿个位数字
   - 对10取模运算得到个位数字
2. 让每一位数字变成个位数字
   - 先除以10，再对10取模得到十位数字
   - 循环上述操作
3. 将每一位数字计算累加
   - 将上次累加结果*10 + 新数字



五. Code 最优解思路及编码实现

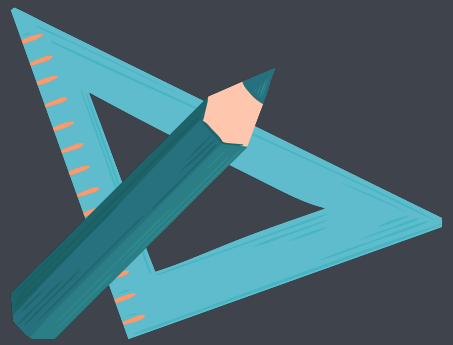
最优解：数学思维解法边界和细节问题

边界问题

- 从低位到高位处理，最高位结束
  - 最高位 / 10 == 0
  - 最高位 % 10 == 最高位
- 数值溢出边界：溢出则返回0
  - 用long类型存放，溢出int则返回0
  - 新整数补充最后一位前判断溢出

细节问题

- 首位不为0
- 符号处理



五. Code 最优解思路及编码实现

拉勾教育
—互联网人实战大学—

```java
public int reverse(int x) {
    if(x == Integer.MAX_VALUE || x == Integer.MIN_VALUE) {
        // 整数类型最小值的绝对值 比 最大值的绝对值 大1
        return 0; // 反转必然溢出，返回0
    }
    int sign = x > 0 ? 1 : -1; // 符号
    x = x < 0 ? x * -1 : x; // 无论正负，都当成正数
    int result = 0; // 返回结果
    int last = 0; // 末位
    // 1.尝试拿个位数字：对10取模运算
    // 2.让每一位数字变成个位数字：先除以10，再对10取模得到十位数字
    while ((last = x % 10) != x) { // 条件不成立：最高位%10==最高位
        // 3.将每一位数字计算累加：将上次累加结果*10 + 新数字
        result = result * 10 + last;
        x /= 10;
    }
    if (last != 0) { // 此时last为最高位，单独处理
        long re = result; // 处理越界
        re = re * 10 + last;
        if (re > Integer.MAX_VALUE || re < Integer.MIN_VALUE)
            result = 0;
        else
            result = (int)re; // 向下转型
    }
    return result * sign; // 返回前进行符号处理
}
```

时间复杂度：O(n)

- 遍历整数的每一位：O(n)

空间复杂度：O(1)

- 只需要一个整数变量O(1)

执行耗时:1 ms,击败了100% 的Java用户
内存消耗:35.4 MB,击败了99.99% 的Java用户

六. Change 变形延伸

拉勾教育
—互联网人实战大学—

题目变形

- （练习）对长整形数据进行反转
- （练习）对字符串进行反转

延伸扩展

- 在面对一些数字类题目时，思考是否可以利用数学特性来解决问题！

本题来源：

- Leetcode 7 https://leetcode-cn.com/problems/reverse-integer/

总结

拉勾教育
—互联网人实战大学—

6C解题法

- Comprehend 理解题意
- Choose 选择数据结构和算法思维
- Code 基本解法编码
- Consider 思考更优解
- Code 最优解编码
- Change 变形延伸

总结

数据结构：数组

- 数组容量固定不变，需在创建数组时指定
- 使用连续的物理空间在存取数据
- 可以通过下标在O(1)的时间复杂度下读取数据

算法思维

- 遍历
- 逆序
- 原地交换
- 数学思维：取模、累加

课后练习

拉勾教育
—互联网人实战大学—

1. 实现一个函数*atoi*，使其能够将字符串转换整数 (Leetcode 8/中等)
2. 颠倒给定的32位无符号整数的二进制位（ Leetcode 190/简单 ）
3. 返回一个整数其二进制表达式中数字位为'1'的个数（ Leetcode 191 /简单 ）
4. 验证给定字符串是否可以解释为十进制数字（ Leetcode 65/困难 ）

拉勾教育

—互联网人实战大学—

下载「拉勾教育App」
获取更多内容