题目描述

给定一个整型数组，请从该数组中选择3个元素组成最小数字并输出

（如果数组长度小于3，则选择数组中所有元素来组成最小数字）。

输入描述

一行用半角逗号分割的字符串记录的整型数组，0 < 数组长度 <= 100，0 < 整数的取值范围 <= 10000。

输出描述

由3个元素组成的最小数字，如果数组长度小于3，则选择数组中所有元素来组成最小数字。

用例

输入	21,30,62,5,31
输出	21305
说明	数组长度超过3，需要选3个元素组成最小数字，21305由21,30,5三个元素组成的数字，为所有组合中最小的数字。

输入	5,21
输出	215
说明	数组长度小于3，选择所有元素来组成最小值，215为最小值。

题目解析

此题可以使用暴力法，求n个数取3个全排列，也就是O(n^3)的时间复杂度，但是题目提示0 < 数组长度 <= 100，这个数据规模很容易超时，因此我们应该想一想更优化的方法。

我们知道Array.prototype.sort默认排序是按照Unicode值从小到大排的，~~因此对于只有两个数的情况，我们直接按照sort字典序升序，比如5,21，字典序升序后就是21,5，而215就是最小组合数。~~

> 2023.02.03 这里直接对数组进行字典序升序，拼接后得到的组合数，不一定是最小的，比如数组 [3, 32, 321]，此时按照字典序升序后，还是 [3, 32, 321]，拼接出来为332321，而这显然不是最小的组合数，最小的组合数应该是321323。
> 此处，得到最小组合数的正确排序规则应该是

对于三个数及以上的数组，我们需要从中取出3个数，这个3个数，首先需要保证总长度最短，即保证组合数的位数最少，其值才能最小，因此我们需要将数组升序，这样小数在前，大数在后，我们只要取前三位即可，比如21,30,62,5,31升序为 5,21,30,31,62，取前3个，5,21,30然后进行sort默认排序，变为21，30，5，而21305就是最小值。

JavaScript算法源码

```javascript
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

rl.on("line", (line) => {
  const strs = line.split(",");
  console.log(getResult(strs));
});

function getResult(strs) {
  strs.sort((a. b) => a - b):

  return strs
    .slice(0, 3)
    .sort((a, b) => {
      const s1 = a + b;
      const s2 = b + a;
      return s1 == s2 ? 0 : s1 > s2 ? 1 : -1;
    })
    .join("");
}
```

Java算法源码

```java
import java.util.Arrays;
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    String[] strs = sc.nextLine().split(",");
    System.out.println(getResult(strs));
  }

  public static String getResult(String[] strs) {
    Arrays.sort(strs, (a, b) -> Integer.parseInt(a) - Integer.parseInt(b));

    String[] tmp = Arrays.copyOfRange(strs, 0, Math.min(3, strs.length));
    Arrays.sort(tmp, (a, b) -> (a + b).compareTo(b + a));

    StringBuilder sb = new StringBuilder();
    for (String s : tmp) {
      sb.append(s);
    }

    return sb.toString();
  }
}
```

Python算法源码

```python
import functools

# 输入获取
strs = input().split(",")


# 算法入口
def cmp(a, b):
    s1 = a + b
    s2 = b + a
    return 0 if s1 == s2 else 1 if s1 > s2 else -1


def getResult(strs):
    strs.sort(key=lambda x: int(x))
    tmp = strs[:3]
    tmp.sort(key=functools.cmp_to_key(cmp))
    return "".join(tmp)


# 算法调用
print(getResult(strs))
```