中国科学院大学
University of Chinese Academy of Sciences

编程基础-2

切片与递归

简版快速排序程序fastsort.go

提纲

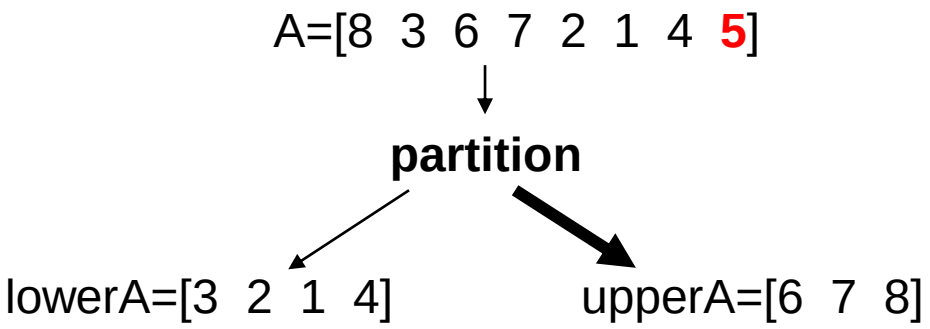
- 目标：开发出fastsort.go排序程序，理解递归与切片

- 步骤
  - 使用8元素例子理解fastsort排序算法
  - 理解fastsort.go程序框架
  - 现场实现课程提供的fastsort.go程序
  - 验证你的fastsort.go程序的正确性
  - 提交你的fastsort.go程序

课件中包含教科书未包括的素材引用，特此致谢

1 课堂上讲的简版快速排序算法fastsort

- **输入**：8元素整数数组d = [8 3 6 7 2 1 4 5]
- **输出**：8元素整数数组d = [1 2 3 4 5 6 7 8]
  - 元素从小到大排好序了
- **步骤**： 调用fastsort(d)。
  函数fastsort(A)的计算过程如下：
  1. 基线情况（base case）：如果A只包含0个元素（如[]）或1个元素（如[3]），fastsort(A)结束
  2. 选择A的最后一个元素作为标杆元素（pivot）
  3. 调用划分函数partition
     - 将小于标杆元素的元素放入lowerA子数组（称为小数组）中，将大于标杆元素的元素放入upperA子数组（称为大数组）中
  4. 递归调用fastsort(lowerA)
  5. 递归调用fastsort(upperA)



d = [8 3 6 7 2 1 4 5]

d = [3 2 1 4] 5 [6 7 8]

d = [3 2 1] 4 [] 5 [6 7 8]

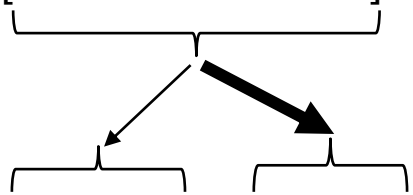
d = [] 1 [2 3] 4 5 [6 7 8]
d = 1 [2] 3 [] 4 5 [6 7 8]
d = 1 [] 2 [] 3 4 5 [6 7 8]
d = 1 2 3 4 5 [6 7] 8 []
d= 1 2 3 4 5 [6] 7 [] 8
d = 1 2 3 4 5 [] 6 [] 7 8
d = 1 2 3 4 5 6 7 8

如何体现递归与切片

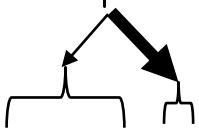
执行过程的简略表示可能引起误解

len(A)<2时，fastsort(A)直接退出

d = [8 3 6 7 2 1 4 **5**]

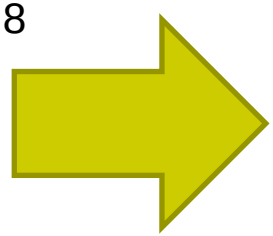


d = [3 2 1 4] **5** [6 7 8]



d = [3 2 1] **4** [] 5 [6 7 8]

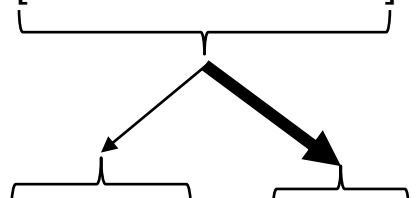
d = [] **1** [2 3] 4 5 [6 7 8]
d = 1 [2] **3** [] 4 5 [6 7 8]
d = 1 [] **2** [] 3 4 5 [6 7 8]
d = 1 2 3 4 5 [6 7] **8** []
d= 1 2 3 4 5 [6] **7** [] 8
d = 1 2 3 4 5 [] **6** [] 7 8
d = 1 2 3 4 5 6 7 8



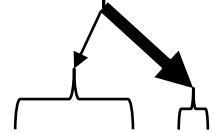
fs([3 2 1 4])
fs([3 2 1])
fs([])
fs([2 3])
fs([2])
fs([])
fs([])
fs([6 7 8])
fs([6 7])
fs([6])
fs([])
fs([])

无省略的执行过程

d = [8 3 6 7 2 1 4 **5**]



d = [3 2 1 **4**] 5 [6 7 8]

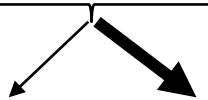


d = [3 2 **1**] 4 [] 5 [6 7 8]
d = [] 1 [2 3] 4 [] 5 [6 7 8]
d = 1 [2 **3**] 4 [] 5 [6 7 8]
d = 1 [2] 3 [] 4 [] 5 [6 7 8]
d = 1 2 3 [] 4 [] 5 [6 7 8]
d = 1 2 3 4 [] 5 [6 7 8]
d = 1 2 3 4 5 [6 7 **8**]
d = 1 2 3 4 5 [6 **7**] 8 []
d= 1 2 3 4 5 [6] 7 [] 8 []
d = 1 2 3 4 5 6 7 [] 8 []
d = 1 2 3 4 5 6 7 8 []
d = 1 2 3 4 5 6 7 8

现场练习：请逐步排序[8 5 6 1 4 3]

- **输入**：6元素整数数组d = [8 5 6 1 4 3]
- **输出**：6元素整数数组d = [1 3 4 5 6 8]
  - 元素从小到大排好序了
- **步骤**： 调用fastsort(d)。
  函数fastsort(A)的计算过程如下：
  1. 基线情况（base case）：如果A只包含0个元素（如[]）或1个元素（如[3]），fastsort(A)结束
  2. 选择A的最后一个元素作为标杆元素（pivot）
  3. 调用划分函数partition
     - 将小于标杆值的元素放入lowerA子数组（称为小数组）中，将大于标杆值的元素放入upperA子数组（称为大数组）中
  4. 递归调用fastsort(lowerA)
  5. 递归调用fastsort(upperA)

d = [8 5 6 1 4 **3**]



d =

d =

d =

d =

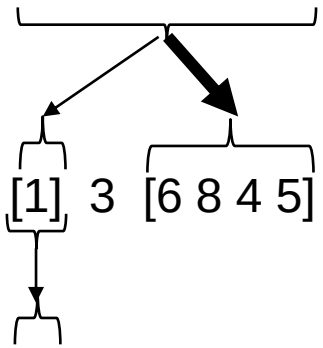
d =

d =

d = 1 3 4 5 6 8

现场练习：请逐步排序[8 5 6 1 4 3]

fs([8 5 6 1 4 3]) d = [8 5 6 1 4 **3**]



fs([1]) d = [1] 3 [6 8 4 5]

fs([6 8 4 5]) d = 1 3 [6 8 4 **5**]

fs([4]) d = 1 3 [4] 5 [6 8]

fs([6 8]) d = 1 3 4 5 [6 **8**]

fs([6]) d = 1 3 4 5 [6] 8 []

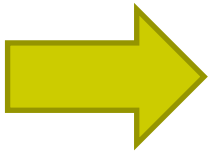
fs([]) d = 1 3 4 5 6 8 []

d = 1 3 4 5 6 8

2. 从算法导出程序fastsort.go

简版快速排序算法

- 输入：8元素整数数组d = [8 3 6 7 2 1 4 5]
- 输出：8元素整数数组d = [1 2 3 4 5 6 7 8]
- 步骤：调用fastsort(d)



- 函数fastsort(A)的计算过程如下：
  1. …

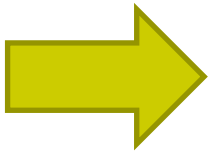
- 函数partition(A)的计算过程如下：
  1. …

```
package main  // fastsort.go
import "fmt"
var d [8]int = [8]int {8,3,6,7,2,1,4,5}
var s []int = d[0:8]
func main() {
   fastsort(s)
   fmt.Println(s)
}

func fastsort(A []int) { … }

func partition(A []int) ([]int, []int) { … }
```

2. 从算法导出程序fastsort.go

简版快速排序算法

- 输入：8元素整数数组d = [8 3 6 7 2 1 4 5]
- 输出：8元素整数数组d = [1 2 3 4 5 6 7 8]
- 步骤：调用fastsort(d)



```
package main  // fastsort.go
import "fmt"
var d [8]int = [8]int {8,3,6,7,2,1,4,5}
var s []int = d[0:8]
func main() {
   fastsort(s)
   fmt.Println(s)
}

func fastsort(A []int) { … }

func partition(A []int) ([]int, []int) { … }
```

- 函数fastsort(A)的计算过程如下：
  1. …

- 函数partition(A)的计算过程如下：
  1. …

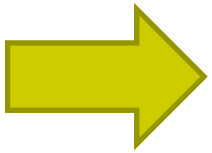
理解切片和数组：

- 在main调用fastsort(s)前，s[0]的值是多少？

2. 从算法导出程序fastsort.go

简版快速排序算法

- 输入：8元素整数数组d = [8 3 6 7 2 1 4 5]
- 输出：8元素整数数组d = [1 2 3 4 5 6 7 8]
- 步骤：调用fastsort(d)



```
package main  // fastsort.go
import "fmt"
var d [8]int = [8]int {8,3,6,7,2,1,4,5}
var s []int = d[0:8]
func main() {
   fastsort(s)
   fmt.Println(s)
}

func fastsort(A []int) { … }

func partition(A []int) ([]int, []int) { … }
```

- 函数fastsort(A)的计算过程如下：
  1. …

- 函数partition(A)的计算过程如下：
  1. …

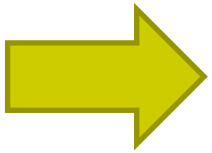
理解切片和数组：

- 执行fastsort(s)之后，d[0]的值是多少？

2. 从算法导出程序fastsort.go

简版快速排序算法

- 输入：8元素整数数组d = [8 3 6 7 2 1 4 5]
- 输出：8元素整数数组d = [1 2 3 4 5 6 7 8]
- 步骤：调用fastsort(d)



```
package main  // fastsort.go
import "fmt"
var d [8]int = [8]int {8,3,6,7,2,1,4,5}
var s []int = d[0:8]
func main() {
   fastsort(s)
   fmt.Println(s)
}

func fastsort(A []int) { … }

func partition(A []int) ([]int, []int) { … }
```

- 函数fastsort(A)的计算过程如下：
  1. …

- 函数partition(A)的计算过程如下：
  1. …

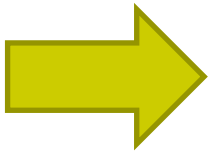
理解切片和数组：

- 若把“fastsort(s)”改成“fastsort(d)”，该程序还能正确执行吗？
  - 若不能，会产生编译错误还是运行时错误？

2. 从算法导出程序fastsort.go

简版快速排序算法

- 输入：8元素整数数组d = [8 3 6 7 2 1 4 5]
- 输出：8元素整数数组d = [1 2 3 4 5 6 7 8]
- 步骤：调用fastsort(d)



```
package main  // fastsort.go
import "fmt"
var d [8]int = [8]int {8,3,6,7,2,1,4,5}
var s []int = d[0:8]
func main() {
   fastsort(s)
   fmt.Println(s)
}

func fastsort(A []int) { … }

func partition(A []int) ([]int, []int) { … }
```

- 函数fastsort(A)的计算过程如下：
  1. …

- 函数partition(A)的计算过程如下：
  1. …

理解切片和数组：

- 若把“fmt.Println(s)”改成“fmt.Println(d)”，程序打印的内容还跟原来相同吗？

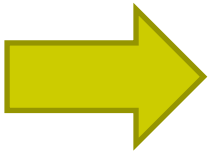
2. 理解fastsort.go

简版快速排序算法

- 输入：8元素整数数组d = [8 3 6 7 2 1 4 5]
- 输出：8元素整数数组d = [1 2 3 4 5 6 7 8]
- 步骤：调用fastsort(d)

- 函数fastsort(A)的计算过程如下：
  1. 基线情况：如果A只包含0个元素（如[]）或1个元素（如[3]），fastsort(A)结束
  2. 选择A的最后一个元素作为标杆元素（pivot）
  3. 调用函数partition(A)
  4. 递归调用fastsort(lowerA)
  5. 递归调用fastsort(upperA)

- 函数partition(A)的计算过程如下：
  1. …



```
package main  // fastsort.go
import "fmt"
var d [8]int = [8]int {8,3,6,7,2,1,4,5}
var s []int = d[0:8]
func main() {
   fastsort(s)
   fmt.Println(s)
}

func fastsort(A []int) {
 if len(A) < 2 {  // 判断是不是基线情况
       return       // 是，退出
   }
   lowerA, upperA := partition(A)
   fastsort(lowerA) // 递归排序小数组
   fastsort(upperA) // 递归排序小数组
}

func partition(A []int) ([]int, []int) { … }
```

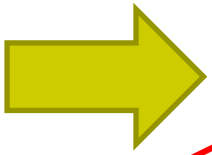
2. 理解fastsort.go

简版快速排序算法

- 输入：8元素整数数组d = [8 3 6 7 2 1 4 5]
- 输出：8元素整数数组d = [1 2 3 4 5 6 7 8]
- 步骤：调用fastsort(d)

- 函数fastsort(A)的计算过程如下：
  1. 基线情况：如果A只包含0个元素（如[]）或1个元素（如[3]），fastsort(A)结束
  2. 选择A的最后一个元素作为标杆元素（pivot）
  3. 调用函数partition(A)
  4. 递归调用fastsort(lowerA)
  5. 递归调用fastsort(upperA)

- 函数partition(A)的计算过程如下：
  1. …



```
package main  // fastsort.go
import "fmt"
var d [8]int = [8]int {8,3,6,7,2,1,4,5}
var s []int = d[0:8]
func main() {
   fastsort(s)
   fmt.Println(s)
}

func fastsort(A []int) {
if len(A) < 2 {  // 判断是不是基线情况
      return       // 是，退出
   }
   lowerA, upperA := partition(A)
   fastsort(lowerA) // 递归排序小数组
   fastsort(upperA) // 递归排序小数组
}

func partition(A []int) ([]int, []int) { … }
```

- 若删除此块代码，程序还能正确打印结果吗？
  - 若不能，会产生编译错误还是运行时错误？

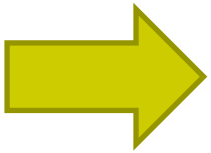
2. 理解fastsort.go

简版快速排序算法

- 输入：8元素整数数组d = [8 3 6 7 2 1 4 5]
- 输出：8元素整数数组d = [1 2 3 4 5 6 7 8]
- 步骤：调用fastsort(d)

- 函数fastsort(A)的计算过程如下：
  1. 基线情况：如果A只包含0个元素（如[]）或1个元素（如[3]），fastsort(A)结束
  2. 选择A的最后一个元素作为标杆元素（pivot）
  3. 调用函数partition(A)
  4. 递归调用fastsort(lowerA)
  5. 递归调用fastsort(upperA)

- 函数partition(A)的计算过程如下：
  1. …



```
package main  // fastsort.go
import "fmt"
var d [8]int = [8]int {8,3,6,7,2,1,4,5}
var s []int = d[0:8]
func main() {
   fastsort(s)
   fmt.Println(s)
}

func fastsort(A []int) {
 if len(A) < 2 {  // 判断是不是基线情况
       return      // 是，退出
   }
   lowerA, upperA := partition(A)
   fastsort(lowerA) // 递归排序小数组
   fastsort(upperA) // 递归排序小数组
}

func partition(A []int) ([]int, []int) { … }
```

- 这两行代码是对lowerA和upperA排序，为什么执行完后A会变得有序？

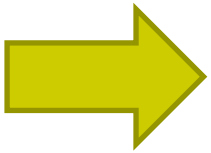
2. 理解fastsort.go

简版快速排序算法

- 输入：8元素整数数组d = [8 3 6 7 2 1 4 5]
- 输出：8元素整数数组d = [1 2 3 4 5 6 7 8]
- 步骤：调用fastsort(d)

- 函数fastsort(A)的计算过程如下：
  1. 基线情况：如果A只包含0个元素（如[]）或1个元素（如[3]），fastsort(A)结束
  2. 选择A的最后一个元素作为标杆元素（pivot）
  3. 调用函数partition(A)
  4. 递归调用fastsort(lowerA)
  5. 递归调用fastsort(upperA)

- 函数partition(A)的计算过程如下：
  1. …



```
package main  // fastsort.go
import "fmt"
var d [8]int = [8]int {8,3,6,7,2,1,4,5}
var s []int = d[0:8]
func main() {
   fastsort(s)
   fmt.Println(s)
}

func fastsort(A []int) {
 if len(A) < 2 {  // 判断是不是基线情况
       return       // 是，退出
   }
   lowerA, upperA := partition(A)
   fastsort(lowerA) // 递归排序小数组
   fastsort(upperA) // 递归排序小数组
}

func partition(A []int) ([]int, []int) { … }
```

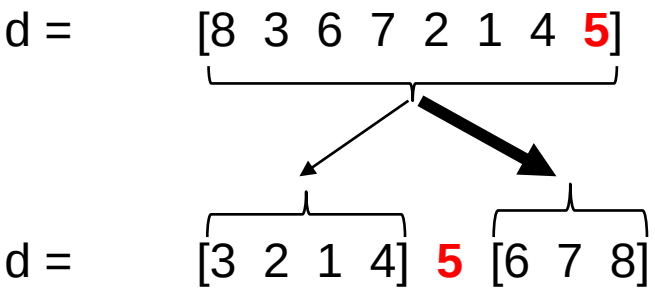
- 为什么fastsort代码中找不到“选择A的最后一个元素作为标杆元素（pivot）” 对应的代码？

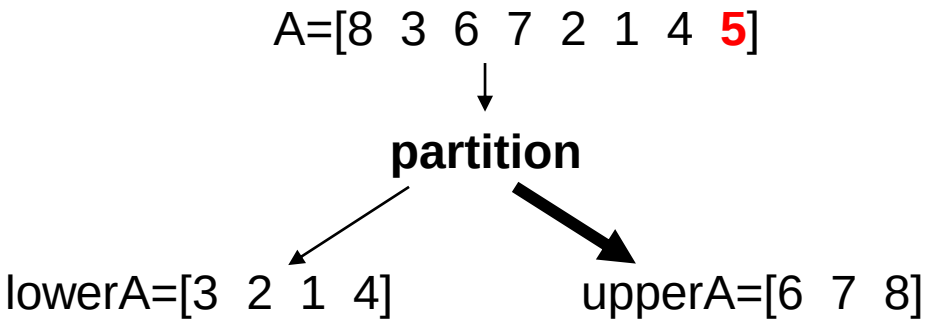
简版快速排序算法fastsort

- **输入**：8元素整数数组d = [8 3 6 7 2 1 4 5]
- **输出**：8元素整数数组d = [1 2 3 4 5 6 7 8]
  - 元素从小到大排好序了
- **步骤**： 调用fastsort(d)。
  函数fastsort(A)的计算过程如下：
  1. 基线情况（base case）
     - 如果A只包含0个或1个元素（如[]，[3]），fastsort(A)结束

  编程假设：选择A的最后一个元素作为标杆元素

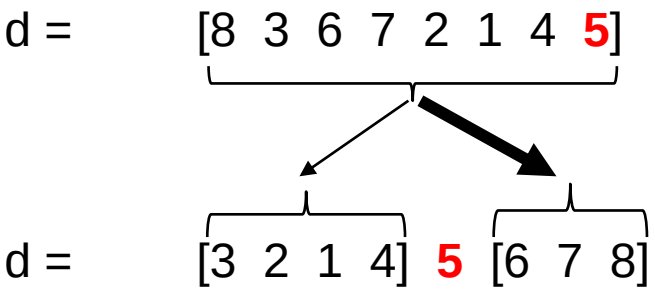
  2. 调用划分函数partition
     - 将小于标杆值的元素放入lowerA子数组（称为小数组）中，
       将大于标杆值的元素放入upperA子数组（称为大数组）中
  3. 递归调用fastsort(lowerA)
  4. 递归调用fastsort(upperA)

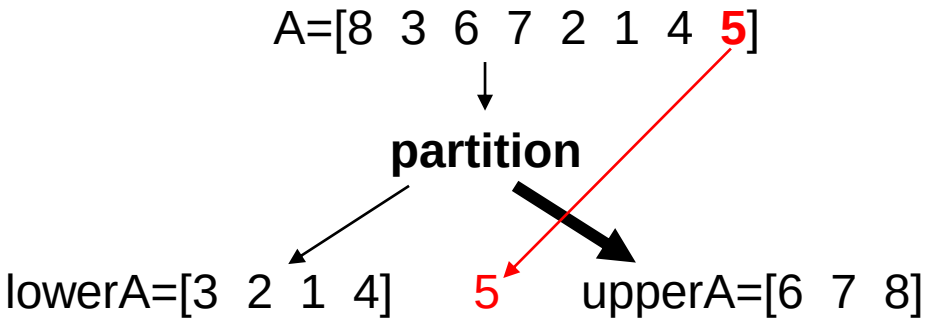




简版快速排序算法fastsort

- **输入**：8元素整数数组d = [8 3 6 7 2 1 4 5]
- **输出**：8元素整数数组d = [1 2 3 4 5 6 7 8]
  - 元素从小到大排好序了
- **步骤**： 调用fastsort(d)。
  函数fastsort(A)的计算过程如下：
  1. 基线情况（base case）
     - 如果A只包含0个或1个元素（如[]，[3]），fastsort(A)结束
  2. 调用划分函数partition
     - 将小于标杆值的元素放入lowerA子数组（称为小数组）中，将大于标杆值的元素放入upperA子数组（称为大数组）中
     - 注意：标杆的位置变了
  3. 递归调用fastsort(lowerA)
  4. 递归调用fastsort(upperA)





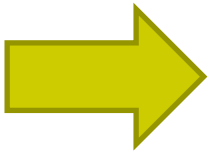
2. 理解fastsort.go

简版快速排序算法

- 输入：8元素整数数组d = [8 3 6 7 2 1 4 5]
- 输出：8元素整数数组d = [1 2 3 4 5 6 7 8]
- 步骤：调用fastsort(d)

- 函数fastsort(A)的计算过程如下：
  1. 基线情况：如果A只包含0个元素（如[]）或1个元素（如[3]），fastsort(A)结束
  2. 选择A的最后一个元素作为标杆元素
  3. 调用函数partition(A)
  4. 递归调用fastsort(lowerA)
  5. 递归调用fastsort(upperA)

- 函数partition(A)的计算过程如下：
  1. 将小于标杆元素的A的元素放入lowerA子数组，将大于标杆元素的A的元素放入upperA子数组
  2. 返回两个子数组lowerA和upperA



```
package main  // fastsort.go
import "fmt"
var d [8]int = [8]int {8,3,6,7,2,1,4,5}
var s []int = d[0:8]
func main() {
   fastsort(s)
   fmt.Println(s)
}

func fastsort(A []int) {
 if len(A) < 2 {  // 判断是不是基线情况
       return          // 是，退出
   }
   lowerA, upperA := partition(A)
   fastsort(lowerA) // 递归排序小数组
   fastsort(upperA) // 递归排序小数组
}

func partition(A []int) ([]int, []int) {  // len(A)>1
   lower := 0  // lower 是lowerA的长度，初始值为0
   for i:= 0; i<len(A); i++ {  // 逐个扫描A的元素A[i]
      // 此处插入你的代码
      // 建议：如果A[i]<标杆值，
      //  A[i]与A[lower]交换并更新lower
   }
   // 此时lower的值是len(lowerA)+1；
   // 交换，使得标杆紧跟在lowerA之后
   A[lower], A[len(A)-1] = A[len(A)-1], A[lower]
   // 返回切片lowerA和upperA
   // 注意两个切片的起始索引和结束索引
   return A[0:lower], A[lower+1:len(A)]
}
```

Partition示例

```
func partition(A []int) ([]int, []int) {  // len(A)>1
   lower := 0  // lower 是lowerA的长度，初始值为0
   for i:= 0; i<len(A); i++ {  // 逐个扫描A的元素A[i]
      // 此处插入你的代码
      // 建议：如果A[i]<标杆值，
      //  A[i]与A[lower]交换并更新lower
   }
   // 此时lower的值是len(lowerA)+1；
   // 交换，使得标杆紧跟在lowerA之后
   A[lower], A[len(A)-1] = A[len(A)-1], A[lower]
   // 返回切片lowerA和upperA
   // 注意两个切片的起始索引和结束索引
   return A[0:lower], A[lower+1:len(A)]
}
```

A = [8 5 6 1 4 3]

A = [1] 3 [6 8 4 5]

0 1 2 3 4 5 索引

i↓

[8 5 6 1 4 3] // lower, i 初始值为0, len(A)-1为5

lower↑ // A[0] > A[5], i = i + 1

i↓

[8 5 6 1 4 3] // A[1] > A[5], i = i + 1

lower↑

i↓

[8 5 6 1 4 3] // A[2] > A[5], i = i + 1

lower↑

i↓

[8 5 6 1 4 3] // A[3] < A[5], A[i]与A[lower]交换

lower↑ // lower = lower + 1, i = i + 1

i↓

[1 5 6 8 4 3] // A[4] > A[5], i = i + 1

lower↑

i↓

[1 5 6 8 4 3] // 循环结束，交换标杆和A[lower]

lower↑

[1 3 6 8 4 5] // lowerA = A[0:lower] = A[0:1] = {1}

// upperA = A[lower+1:len(A)]

// = A[2:6] = {6 8 4 5}

lower↑

3. 现场实现 你的fastsort.go

- 建议：利用右侧代码框架
  - 只在标红处填充你的代码

验证

- 当d = [8 3 6 7 2 1 4 5]时，计算出正确结果d = [1 2 3 4 5 6 7 8]

- 可以重写整个partition函数体

```
package main  // fastsort.go
import "fmt"
var d [8]int = [8]int {8,3,6,7,2,1,4,5}
var s []int = d[0:8]
func main() {
   fastsort(s)
   fmt.Println(s)
}

func fastsort(A []int) {
 if len(A) < 2 {  // 判断是不是基线情况
       return       // 是，退出
   }
   lowerA, upperA := partition(A)
   fastsort(lowerA) // 递归排序小数组
   fastsort(upperA) // 递归排序小数组
}

func partition(A []int) ([]int, []int) {
   lower := 0
   for i:= 0; i<len(A); i++ {
      // 此处插入你的代码
      // 建议：如果A[i]<标杆值，
      //  A[i]与A[lower]交换并更新lower
   }
   A[lower], A[len(A)-1] = A[len(A)-1], A[lower]
   return A[0:lower], A[lower+1:len(A)]
}
```

验证fastsort.go

验证-1

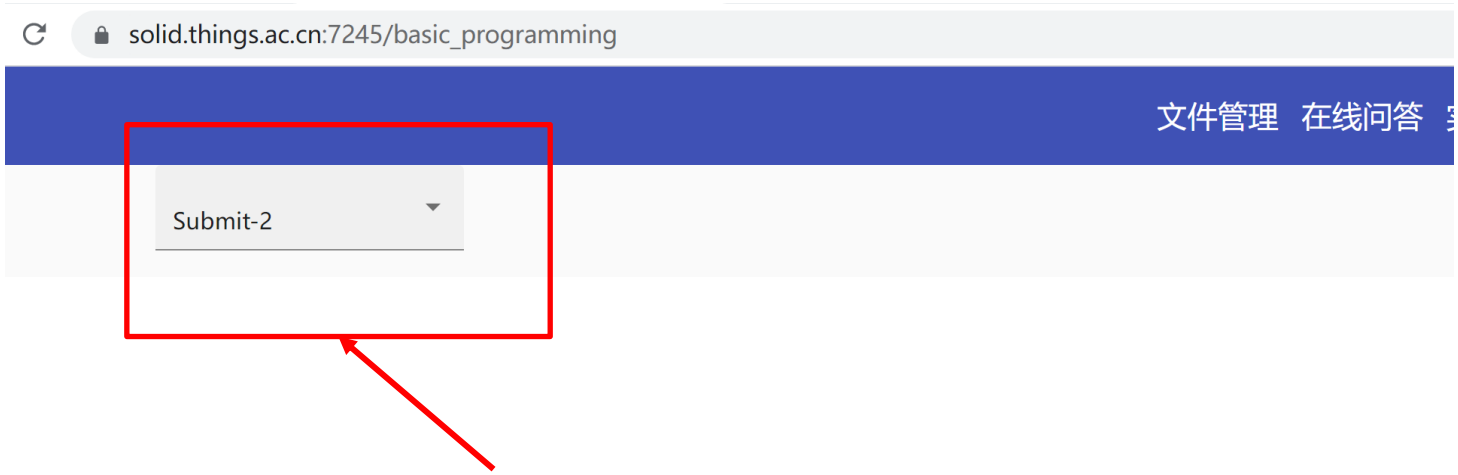
- 当d = [8 3 6 7 2 1 4 5]时，计算出正确结果d = [1 2 3 4 5 6 7 8]

- 修改程序，验证-2
  - 当d = [8 5 6 7 2 1 4 3]时，计算出正确结果d = [1 2 3 4 5 6 7 8]

- 修改程序，验证-3
  - 当d = [8 3 6 7 2 1]时，计算出正确结果d = [1 2 3 6 7 8]

下课前提交你的fastsort.go程序

请提交验证-1、验证-2和验证-3 三个程序源码。
代码文件提交链接 https://www.solid.things.ac.cn:7245/basic_programming



选择Submit-2 （第2次课的编程练习）