Golang sort库——排序

## [func Find(n int, cmp func(int) int) (i int, found bool)](https://pkg.go.dev/sort@go1.19.4" \l "Find) added in go1.19

## [func Float64s(x []float64)](https://pkg.go.dev/sort@go1.19.4" \l "Float64s)：给float64数组升序排序

## [func Float64sAreSorted(x []float64) bool](https://pkg.go.dev/sort@go1.19.4" \l "Float64sAreSorted)：判断float64数组是否已升序排序

## [func Ints(x []int)](https://pkg.go.dev/sort@go1.19.4" \l "Ints)：给int数组升序排序

## [func IntsAreSorted(x []int) bool](https://pkg.go.dev/sort@go1.19.4" \l "IntsAreSorted)：判断int数组是否已升序排序

## [func IsSorted(data Interface) bool](https://pkg.go.dev/sort@go1.19.4" \l "IsSorted)：变量是否已排序

## [func Search(n int, f func(int) bool) int](https://pkg.go.dev/sort@go1.19.4" \l "Search)：从数组中搜索满足f函数的最近一个索引

var data = []int{1, 3, 654, 676}  
x := 23  
i := sort.Search(len(data), func(i int) bool { return data[i] >= x })  
if i < len(data) && data[i] == x {  
 // x is present at data[i]  
} else {  
 // x is not present in data,  
 // but i is the index where it would be inserted.  
}

## [func SearchFloat64s(a []float64, x float64) int](https://pkg.go.dev/sort@go1.19.4" \l "SearchFloat64s)：在float64数组a中找到等于x的元素的索引

a := []float64{1.0, 2.0, 3.3, 4.6, 6.1, 7.2, 8.0}  
  
x := 2.0  
i := sort.SearchFloat64s(a, x)  
fmt.Printf("found %g at index %d in %v\n", x, i, a) //found 2 at index 1 in [1 2 3.3 4.6 6.1 7.2 8]  
  
x = 0.5  
i = sort.SearchFloat64s(a, x)  
fmt.Printf("%g not found, can be inserted at index %d in %v\n", x, i, a)//0.5 not found, can be inserted at index 0 in [1 2 3.3 4.6 6.1 7.2 8]

## [func SearchInts(a []int, x int) int](https://pkg.go.dev/sort@go1.19.4" \l "SearchInts)：在int数组a中找到等于x的元素的索引

## [func SearchStrings(a []string, x string) int](https://pkg.go.dev/sort@go1.19.4" \l "SearchStrings)：在string数组a中找到等于x的元素的索引

## [func Slice(x any, less func(i, j int) bool)](https://pkg.go.dev/sort@go1.19.4" \l "Slice)：根据迭代less函数规则来进行自定义排序

## [func SliceIsSorted(x any, less func(i, j int) bool) bool](https://pkg.go.dev/sort@go1.19.4" \l "SliceIsSorted)：判断是否根据迭代less函数规则来进行自定义排序

## [func SliceStable(x any, less func(i, j int) bool)](https://pkg.go.dev/sort@go1.19.4" \l "SliceStable)：根据迭代less函数规则来进行自定义排序（使用稳定排序）

## [func Sort(data Interface)](https://pkg.go.dev/sort@go1.19.4" \l "Sort)：给数组data排序

## [func Stable(data Interface)](https://pkg.go.dev/sort@go1.19.4" \l "Stable)：给数组data稳定排序

## [func Strings(x []string)](https://pkg.go.dev/sort@go1.19.4" \l "Strings)：给字符串数组x升序排序

## [func StringsAreSorted(x []string) bool](https://pkg.go.dev/sort@go1.19.4" \l "StringsAreSorted)：判断字符串数组x是否升序排序

# [type Float64Slice](https://pkg.go.dev/sort@go1.19.4" \l "Float64Slice)

## [func (x Float64Slice) Len() int](https://pkg.go.dev/sort@go1.19.4" \l "Float64Slice.Len)

## [func (x Float64Slice) Less(i, j int) bool](https://pkg.go.dev/sort@go1.19.4" \l "Float64Slice.Less)

## [func (p Float64Slice) Search(x float64) int](https://pkg.go.dev/sort@go1.19.4" \l "Float64Slice.Search)

## [func (x Float64Slice) Sort()](https://pkg.go.dev/sort@go1.19.4" \l "Float64Slice.Sort)

## [func (x Float64Slice) Swap(i, j int)](https://pkg.go.dev/sort@go1.19.4" \l "Float64Slice.Swap)

# [type IntSlice](https://pkg.go.dev/sort@go1.19.4" \l "IntSlice)

## [func (x IntSlice) Len() int](https://pkg.go.dev/sort@go1.19.4" \l "IntSlice.Len)

## [func (x IntSlice) Less(i, j int) bool](https://pkg.go.dev/sort@go1.19.4" \l "IntSlice.Less)

## [func (p IntSlice) Search(x int) int](https://pkg.go.dev/sort@go1.19.4" \l "IntSlice.Search)

## [func (x IntSlice) Sort()](https://pkg.go.dev/sort@go1.19.4" \l "IntSlice.Sort)

## [func (x IntSlice) Swap(i, j int)](https://pkg.go.dev/sort@go1.19.4" \l "IntSlice.Swap)

# [type Interface](https://pkg.go.dev/sort@go1.19.4" \l "Interface)

// An implementation of Interface can be sorted by the routines in this package.  
// The methods refer to elements of the underlying collection by integer index.  
type Interface interface {  
 // Len is the number of elements in the collection.  
 Len() int  
  
 // Less reports whether the element with index i  
 // must sort before the element with index j.  
 //  
 // If both Less(i, j) and Less(j, i) are false,  
 // then the elements at index i and j are considered equal.  
 // Sort may place equal elements in any order in the final result,  
 // while Stable preserves the original input order of equal elements.  
 //  
 // Less must describe a transitive ordering:  
 // - if both Less(i, j) and Less(j, k) are true, then Less(i, k) must be true as well.  
 // - if both Less(i, j) and Less(j, k) are false, then Less(i, k) must be false as well.  
 //  
 // Note that floating-point comparison (the < operator on float32 or float64 values)  
 // is not a transitive ordering when not-a-number (NaN) values are involved.  
 // See Float64Slice.Less for a correct implementation for floating-point values.  
 Less(i, j int) bool  
  
 // Swap swaps the elements with indexes i and j.  
 Swap(i, j int)  
}

## [func Reverse(data Interface) Interface](https://pkg.go.dev/sort@go1.19.4" \l "Reverse)

# [type StringSlice](https://pkg.go.dev/sort@go1.19.4" \l "StringSlice)

## [func (x StringSlice) Len() int](https://pkg.go.dev/sort@go1.19.4" \l "StringSlice.Len)

## [func (x StringSlice) Less(i, j int) bool](https://pkg.go.dev/sort@go1.19.4" \l "StringSlice.Less)

## [func (p StringSlice) Search(x string) int](https://pkg.go.dev/sort@go1.19.4" \l "StringSlice.Search)

## [func (x StringSlice) Sort()](https://pkg.go.dev/sort@go1.19.4" \l "StringSlice.Sort)

## [func (x StringSlice) Swap(i, j int)](https://pkg.go.dev/sort@go1.19.4" \l "StringSlice.Swap)