作业 1.2

题目描述

> 给定一个升序数组，找出所有`A[i] = i`的下标，并分析其复杂度

算法描述、证明与时间复杂度分析

- 由于原序列单调增，且两两不同
- 当原序列中仅有整数时
  - $a[i+1]-a[i] \geq (i+1)-(i)$
  - $\therefore if\ a[i] > i\ then\ a[i+k] > i+k$
    $if\ a[i] < i\ then\ a[i+k] < i+k$
  - 因此只需要找到两端点位置
    - 采用二分法，分别获取端点位置
    - 根据 $(*)$ 式，先找到任意 t , 令 $a[t]=t$, $F(N)=O(logN)$
    - 分别以t为右端点和左端点,寻找合法区间的左端点和右端点,采用二分法, $F'(N)=O(logN)$
  - 时间复杂度
    - 算法的时间复杂度为 $F(N)=O(logN)$
    - 如果考虑数据读入时间，则读入耗时最久，为$F(N)=O(N)$
- 当原序列存在小数时
  - 此时，最好的算法是扫描整个序列并记录，时间复杂度为$F(N)=O(N)$

代码

```golang
//golang
var N int
var data []int

func getLBound(l, r int) (ll int) {
        for l <= r {
                mid := (l + r) >> 1
                if mid == data[mid] {
                        ll = getLBound(l, mid-1)
                        if ll == -1 {
                                ll = mid
                        }
                        return
                }
                if mid > data[mid] {
                        l = mid + 1
                } else {
                        r = mid - 1
                }
        }
```

```
        ll = -1
        return
}
func getRBound(l, r int) (rr int) {
        for l <= r {
                mid := (l + r) >> 1
                if mid == data[mid] {
                        rr = getRBound(mid+1, r)
                        if rr == -1 {
                                rr = mid
                        }
                        return
                }
                if mid > data[mid] {
                        l = mid + 1
                } else {
                        r = mid - 1
                }
        }
        rr = -1
        return
}

func main() {
        fmt.Scanln(&N)
        data = make([]int, N+1)
        for i, _ := range data[1 : N+1] {
                fmt.Scanf("%d", &data[i+1])
        }
        l, r := getLBound(1, N), getRBound(1, N)

        fmt.Printf("ANS = [%d, %d]", l, r)
}
```

- 运行结果

```
8
-1 2 3 4 5 6 7 8
ANS = [2, 6]
```

```
9
1 2 3 4 5 6 7 8 9
ANS = [1, 9]
```