



如何进行快速的查找



张华

64174234@qq.com

内容

1.1 如何从大量的数据集合中查找想要的数据库

1.2 一个简单的学生成绩录入与查询系统

1.3 总结

1.4 思考

1.1如何从大量的数据集合中查找想要的数

- 我们编写计算机软件系统的过程中，增、删、改、查是最常用的四大操作，这四大操作尤其以查找操作最为频繁
 - 增加数据的时候先要找到插入的位置，然后再插入
 - 删除数据的时候先要找到删除的位置，然后再删除
 - 修改数据的时候先要找到修改的位置，然后再修改
- 在计算机科学中查找定义为：在一些（有序的/无序的）数据元素中，通过一定的方法找出与给定关键字相同的数据元素的过程叫做查找。也就是根据给定的某个值，在查找数据集中确定一个关键字等于给定值的记录或数据元素。

1.1如何从大量的数据集合中查找想要的数

□ 查找算法哪家强？

序号	姓名	期末成绩
1	刘雨星	98
2	陶明明	76
3	宋文杰	85
4	王玉	73
5	葛嘉莉	66
6	王浩	87
7	李志鹏	78
8	林凯	99
9	刘家威	100
10	郭丰瑞	97
11	李荣东	85
12	蒋之凯	69
13	吴卓	96
14	张韬	76
15	仇单宁	67
...

无序表，对成绩字段查找
需要逐个比较，复杂度为 $O(n)$

序号	姓名	期末成绩
1	葛嘉莉	66
2	仇单宁	67
3	蒋之凯	69
4	王玉	73
5	陶明明	76
6	张韬	76
7	李志鹏	78
8	宋文杰	85
9	李荣东	85
10	王浩	87
11	吴卓	96
12	郭丰瑞	97
13	刘雨星	98
14	林凯	99
15	刘家威	100
...

有序表，对成绩字段查找
可采用折半查找，复杂度为 $O(\log_2 n)$

□ 是否存在“1次”就能找到的方法？即复杂度为 $O(1)$

□ 采用哈希表结构

1.1如何从大量的数据集合中查找想要的数

- 哈希表—本质是映射 $pos=f(key) \quad key \in data$
- 例如，有若干条数据，只有一个属性，当然也是关键字，即 $key=data$

- {Zhao, Qian, Sun, Li, Wu, Chen, Han, Ye, Dei }

- 将这些数据装填到如下所示的哈希表的存储空间中

0	1	2	3	4	5	6	7	8	9	10	11	12	13
	Chen	Dei		Han		Li		Qian	Sun		Wu	Ye	Zhao

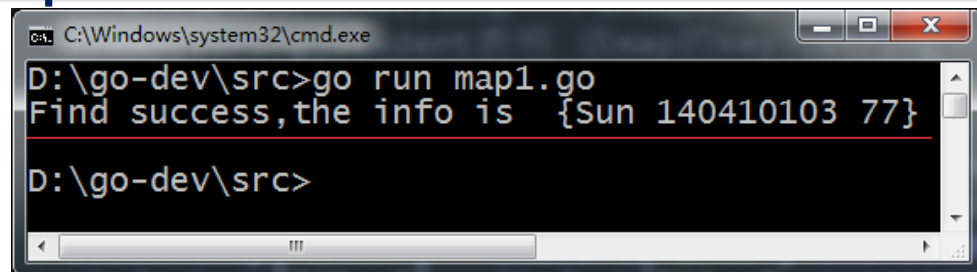
- 以 $val=(\text{firstletter}(\text{key})-\text{'A'}+1)/2$ 作为映射函数进行装填
 - 这种情况下的查找变得很简单，只要给出一个查找关键字 key ，就能计算出其在哈希表中的位置，时间复杂度是 $O(1)$ ，而不是如前面介绍的方法那样，需要进行多次比较才能找到关键字
- 后续《数据结构》课程对哈希表会有更详细的介绍

1.1如何从大量的数据集合中查找想要的数

□ Go中的哈希表类型map

```
package main
import "fmt"
type Student struct { //定义学生成绩类型
    name string //姓名
    stuno int    //学号
    score int    //成绩
}
```

```
func main() {
    //用内置函数make()创建一个新map, 名称为studentmap,
    //map[string]Student表明 该map的key为string类型,存放的数据为Student类型
    //我们将这个string类型的key对应map中的姓名字段
    studentmap := make(map[string]Student)
    //装填数据即为哈希表进行元素赋值,装填函数go map已经为我们封装好,直接使用即可
    //关键字Zhao对应的pos中 放入元素Student{"Zhao", 140410101, 99}
    studentmap["Zhao"] = Student{"Zhao", 140410101, 99}
    studentmap["Qian"] = Student{"Qian", 140410102, 88}
    studentmap["Sun"] = Student{"Sun", 140410103, 77}
    studentmap["Li"] = Student{"Li", 140410104, 66}
    //查找姓名为Sun的成绩, 查找的具体过程go map也已经封装好,直接使用即可
    stuinfo, ok := studentmap["Sun"] //stuinfo代表找到的值,ok代表查找成功与否
    if ok {
        fmt.Println("Find success,the info is ", stuinfo)
    } else {
        fmt.Println("Find fail")
    }
}
```



```
C:\Windows\system32\cmd.exe
D:\go-dev\src>go run map1.go
Find success,the info is {Sun 140410103 77}
D:\go-dev\src>
```

1.2 一个简单的学生成绩录入与查询系统

- 一个班级有30个学生学习Go语言，请完成
 - 录入：录入每个学生的学号、姓名和Go语言成绩
 - 查询：输入学号，找出对应的姓名和成绩
 - 要求用map类型实现

- 量化(数据结构)

- ```
type Student struct{//定义学生对象的类型
 name string//姓名
 stuno int//学号
 score int//成绩
}
```

- ```
studentmap := make(map[string]Student)//学生map
```

- 算法

- ```
studentmap[stuno]//传递学号给studentmap进行查找
```

# 1.2一个简单的学生成绩录入与查询系统(续)

## □ 实现1——非模块化(主函数)

```
func main() {
 studentmap := make(map[int]Student) //建立studentmap, 关键字为int类型
 stu := Student{} //stu接收录入的学生信息, 装填进studentmap
 stunum := 0 //要查找的学号
 n := 0 //学生数量
 fmt.Println("please input the number of students")
 fmt.Scanln(&n) //录入学生数量
 for i := 0; i < n; i++ { //装填studentmap
 fmt.Println("input name of student ", i+1)
 fmt.Scanln(&stu.name) //录入姓名
 fmt.Println("input stuno of student ", i+1)
 fmt.Scanln(&stu.stuno) //录入学号
 fmt.Println("input score of student ", i+1)
 fmt.Scanln(&stu.score) //录入成绩
 studentmap[stu.stuno] = stu //以学号为关键字装填stu到studentmap
 }
 fmt.Println("input the stuno for searching")
 fmt.Scanln(&stunum) //输入查找的学号
 stuinfo, ok := studentmap[stunum] //查找, stuinfo代表找到的值, ok代表查找状态
 if ok { //找到
 fmt.Println("Find success,the info is ", stuinfo)
 } else { //没找到
 fmt.Println("Find fail")
 }
}
```



# 1.2一个简单的学生成绩录入与查询系统(续)

## □ 实现1——非模块化(学生数据类型定义)

```
package main
import "fmt"
type Student struct { //定义学生成绩类型
 name string //姓名
 stuno int //学号
 score int //成绩
}
```

```
D:\go-dev\src>go run map2.go
please input the number of students
3
input name of student 1
zhao
input stuno of student 1
001
input score of student 1
99
input name of student 2
qian
input stuno of student 2
002
input score of student 2
88
input name of student 3
sun
input stuno of student 3
003
input score of student 3
77
input the stuno for searching
002
Find success,the info is {qian 2 88}
```

## 1.2 一个简单的学生成绩录入与查询系统(续)

### □ 实现2——模块化（录入模块）

```
func Input(stumap map[int]Student) { //录入数据，以map作为参数
 stu := Student{} //stu接收录入的学生信息，装填进stumap
 n := 0 //学生数量
 fmt.Println("please input the number of students")
 fmt.Scanln(&n) //录入学生数量
 for i := 0; i < n; i++ { //装填studentmapmap
 fmt.Println("input name of student ", i+1)
 fmt.Scanln(&stu.name) //录入姓名
 fmt.Println("input stuno of student ", i+1)
 fmt.Scanln(&stu.stuno) //录入学号
 fmt.Println("input score of student ", i+1)
 fmt.Scanln(&stu.score) //录入成绩
 stumap[stu.stuno] = stu //以学号为关键字装填stu到studentmap
 }
}
```

# 1.2一个简单的成绩录入与查询系统(续)

## □ 实现2——模块化（查找模块）

```
func Search(stumap map[int]Student) { //查找方法
 stunum := 0 //要查找的学号
 fmt.Println("input the stuno for searching")
 fmt.Scanln(&stunum) //输入查找的学号
 stuinfo, ok := stumap[stunum] //查找, stuinfo代表找到的值, ok代表查找状态
 if ok { //找到
 fmt.Println("Find success,the info is ", stuinfo)
 } else { //没找到
 fmt.Println("Find fail")
 }
}
```

# 1.2一个简单的学生成绩录入与查询系统(续)

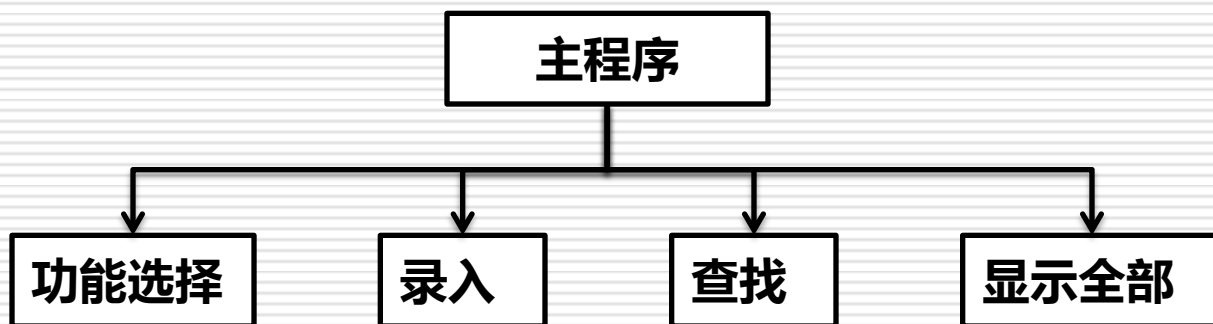
## □ 实现2——模块化（主模块）

```
package main
import "fmt"
type Student struct { //定义学生成绩类型
 name string //姓名
 stuno int //学号
 score int //成绩
}
func main() {
 studentmap := make(map[int]Student) //建立studentmap, 关键字为int类型
 Input(studentmap) //map变量是引用语义, 传递的是地址
 Search(studentmap)
}
```

## 1.2 一个简单的学生成绩录入与查询系统(续)

### □ 实现3——系统化

- 录入模块、查找模块没变
- 增加了显示全部模块
- 增加了菜单定义模块即功能选择模块
- 主模块首先调用功能选择模块



## 1.2 一个简单的学生成绩录入与查询系统(续)

### □ 实现3——系统化（显示全部）

```
func Display(stumap map[int]Student) { //显示
 for _, v := range stumap {
 fmt.Println(v)
 }
}
```

## 1.2 一个简单的学生成绩录入与查询系统(续)

### □ 实现3——系统化（菜单模块）

```
func menu(stumap map[int]Student) {
 for {
 fmt.Println("1-----录入数据")
 fmt.Println("2-----按学号查找")
 fmt.Println("3-----显示全部")
 fmt.Println("4-----退出")
 var s int
 fmt.Scanln(&s)
 switch {
 case s == 1:
 Input(stumap) //录入
 case s == 2:
 Search(stumap) //查找
 case s == 3:
 Display(stumap) //显示全部
 case s == 4:
 os.Exit(0) //退出
 default:
 }
 }
}
```

# 1.2 一个简单的学生成绩录入与查询系统(续)

## □ 实现3——系统化（主模块）

```
package main
import "fmt"
import "os"
type Student struct { //定义学生成绩类型
 name string //姓名
 stuno int //学号
 score int //成绩
}

func main() {
 studentmap := make(map[int]Student) //建立studentmap, 关键字为int类型
 menu(studentmap)
}
```



# 1.3总结

---

## □ map变量的定义

- `var stumap map[int]Student` 定义一个空map变量，空值为 `nil`，多用于函数参数的定义

- `int` 是查找关键字的类型
- `Student` 是map中存放的值的类型

## □ 用make定义map变量

- `stumap:=make(map[int]Student)`

- 可以在创建时候指定初始存储能力

- `stumap:=make(map[int]Student,100)`

- 创建并初始化

- `stumap:=make(map[int]Student){ "140410101" :Student{name:" zhou" ,stuno:140140101,score:99}}`

## □ 元素赋值

- `stumap[140410101]=Student{" zhou" ,140140101,99}}`

# 1.3总结(续)

---

## □ map元素查找

- `stinfo, ok := stumap[140410101]`

- 以140410101为关键字查找，查找结果stinfo代表找到的值,ok代表查找状态

- `if ok { //找到`

- `fmt.Println( "Find success,the info is " ,  
stinfo)//输出找到的对象值`

- `} else { //没找到`

- `fmt.Printf("Find fail\n")`

- `}`

## □ 元素删除

- `delete(stumap,140410101)`

- 删除key为140410101的对象，如果stumap为nil将报错

## 1.3总结(续)

- map对象作为函数参数传递的是地址
- for 中的range 结构 有两个返回结果一个是i，代表在集合中的索引，一个是v，代表i索引对应的元素值
  - 下面代码中没有用到i，就用\_代表用不到，但不能不写，否则报错

```
func Display(stumap map[int]Student) { //显示
 for _, v := range stumap {
 fmt.Println(v)
 }
}
```

# 1.4思考

## □ 在1.2问题的系统化版本基础上

- 增加排序模块，对应menu中s==4分支，退出变为s==5分支
- 增加清屏功能，在菜单模块的最开始处先清屏，即先清掉前面的录入，界面中只显示菜单功能清单（选做）
- 增加按任意键继续功能，在每个功能调用之后立即调用该功能（选做）

```
func menu(stumap map[int]Student) {
 for {
 Clsscr() //显示菜单前调用清屏函数
 fmt.Println("1-----录入数据")
 fmt.Println("2-----按学号查找")
 fmt.Println("3-----显示全部")
 fmt.Println("4-----退出")
 var s int
 fmt.Scanln(&s)
 switch {
 case s == 1:
 Input(stumap) //录入
 Goon() //暂停or按任意键继续
 case s == 2:
 Search(stumap) //查找
 Goon() //暂停or按任意键继续
 case s == 3:
 Display(stumap)
 Goon() //暂停or按任意键继续
 case s == 4:
 os.Exit(0) //退出
 default:
 }
 }
}
```

---

**Thank you very much**

*Any comments and suggestions  
are beyond welcome*