**4.1 实验目的**

加深对于操作系统设备管理技术的了解，体验磁盘移臂调度算法的重要性；掌握几种重要的磁盘移臂调度算法，练习模拟算法的编程技巧，锻炼研究分析试验数据的能力。

**4.2 实验说明**

1. 示例实验程序中模拟两种磁盘移臂调度算法：SSTF算法和SCAN算法

2. 能对两种算法给定任意序列不同的磁盘请求序列，显示响应磁盘请求的过程。

3. 能统计和报告不同算法情况下响应请求的顺序、移臂的总量。

# 实验结果

1. 初始界面



1. 输入磁盘请求序列（可以单个输入，也可以按照[1,2,3,4]形式输入，英文逗号，输入除了数字外的其他字符自动略过）

输入单个数字



输入序列



输入除了数字外的其他字符，无反应



1. 删除磁盘请求号

单击，可以看到磁盘号6被删除



当全部删除时，显示默认文字“输入磁盘请求序列”



1. 输入开始磁盘号，默认为0，输入除了数字外的其他字符无效

输入200，显示“开始序号：200”



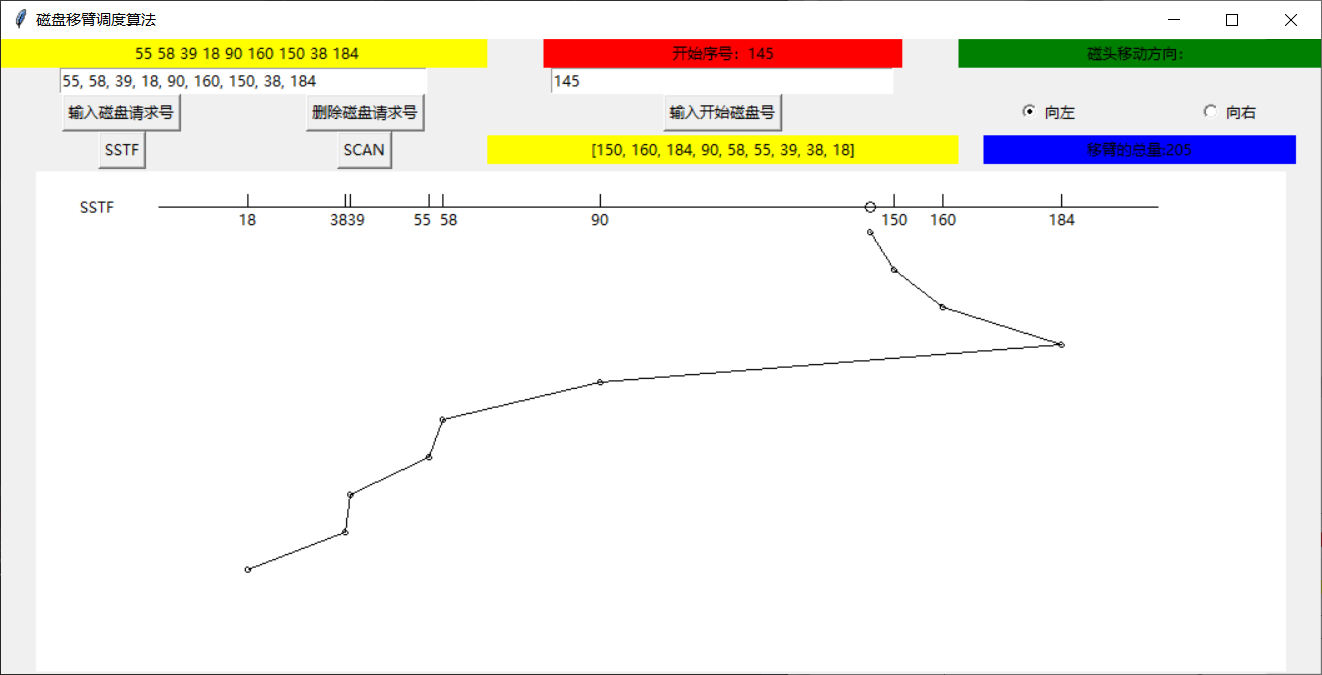
1. 设置磁头移动方向，默认向左（数字变小方向）

点击“向右”，显示“数字变大方向”

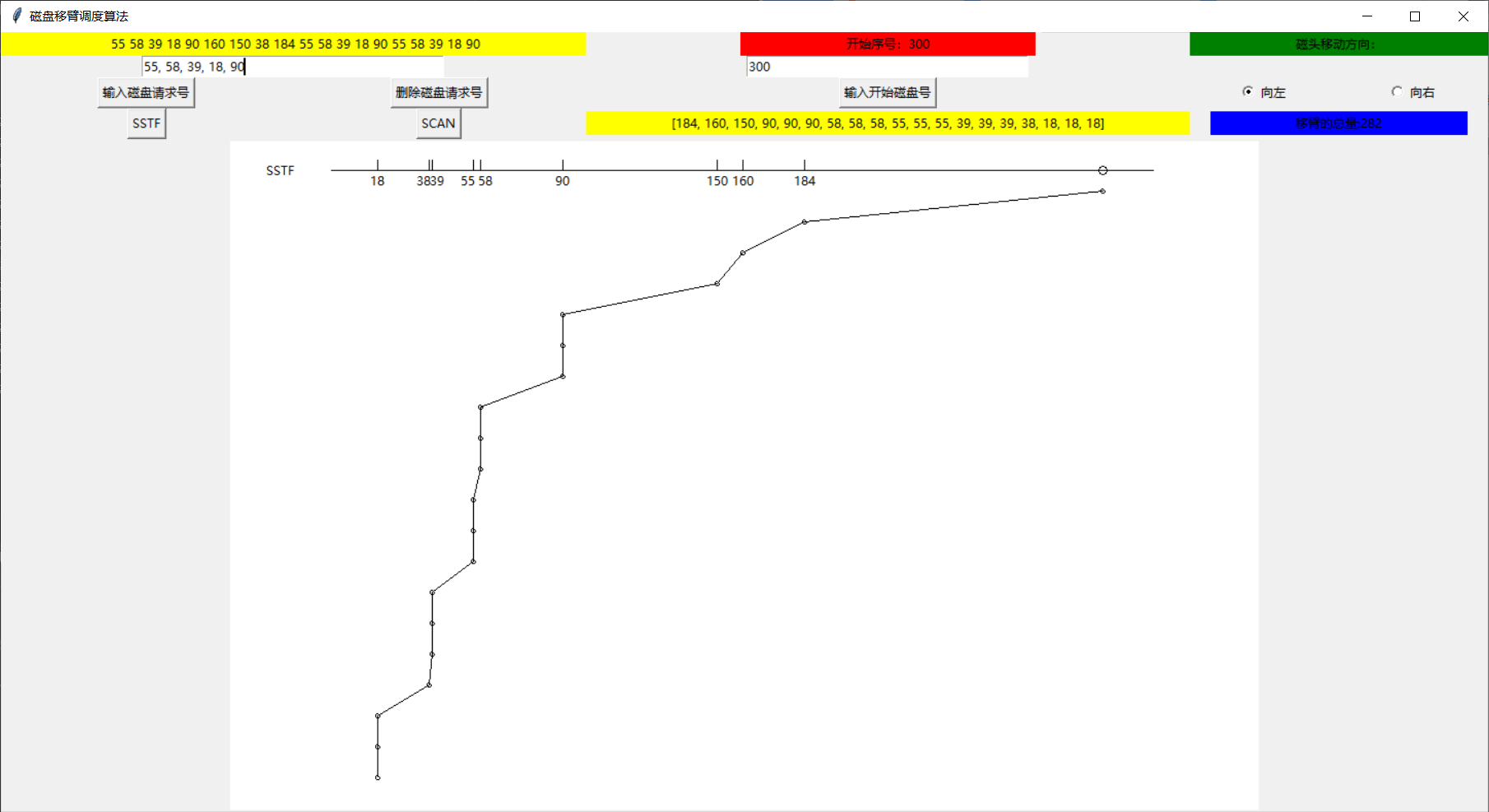


1. 点击SSTF或SCAN，显示算法实施过程，圆圈表示开始序号

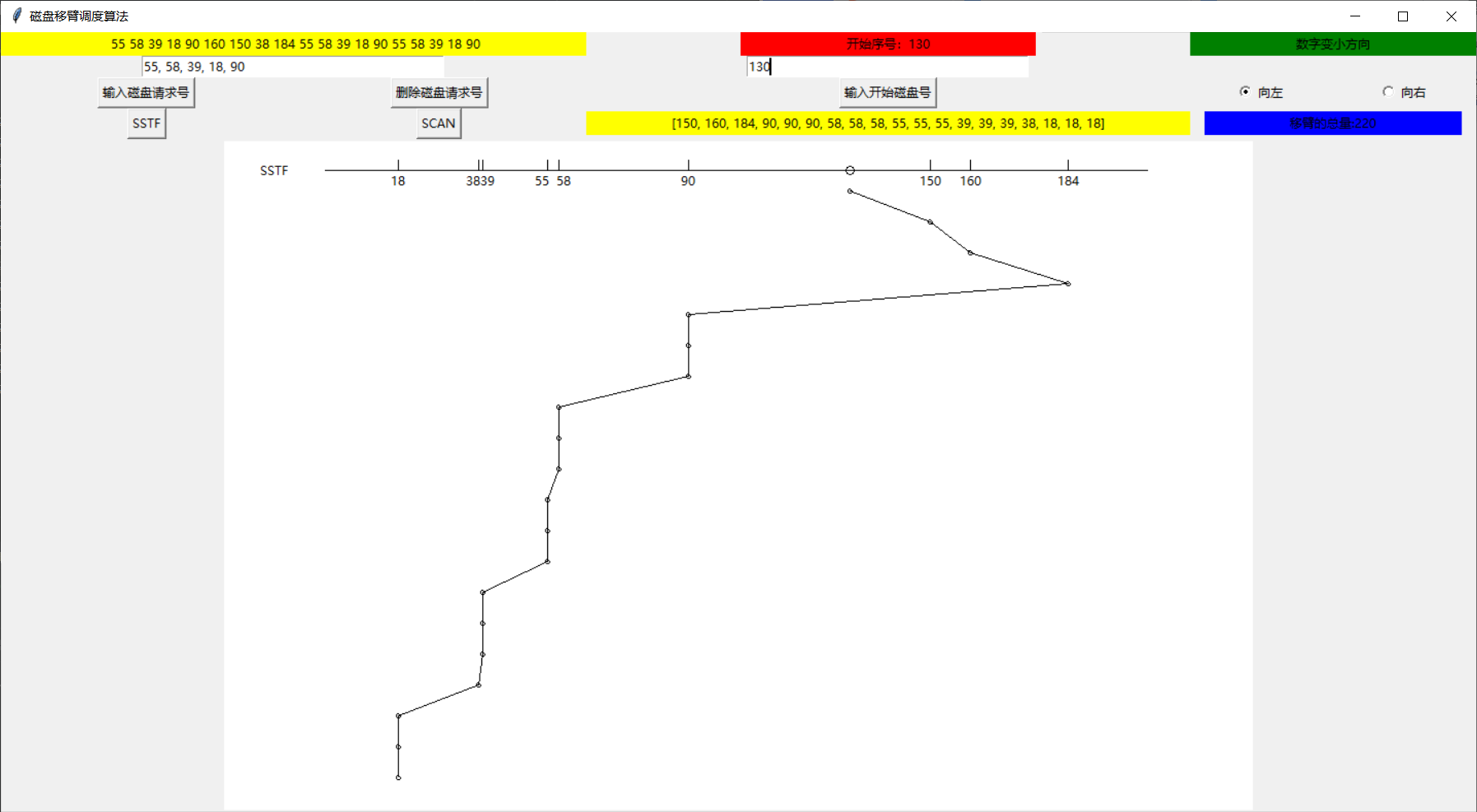
当数字相离太近，自动左右移动，如3839

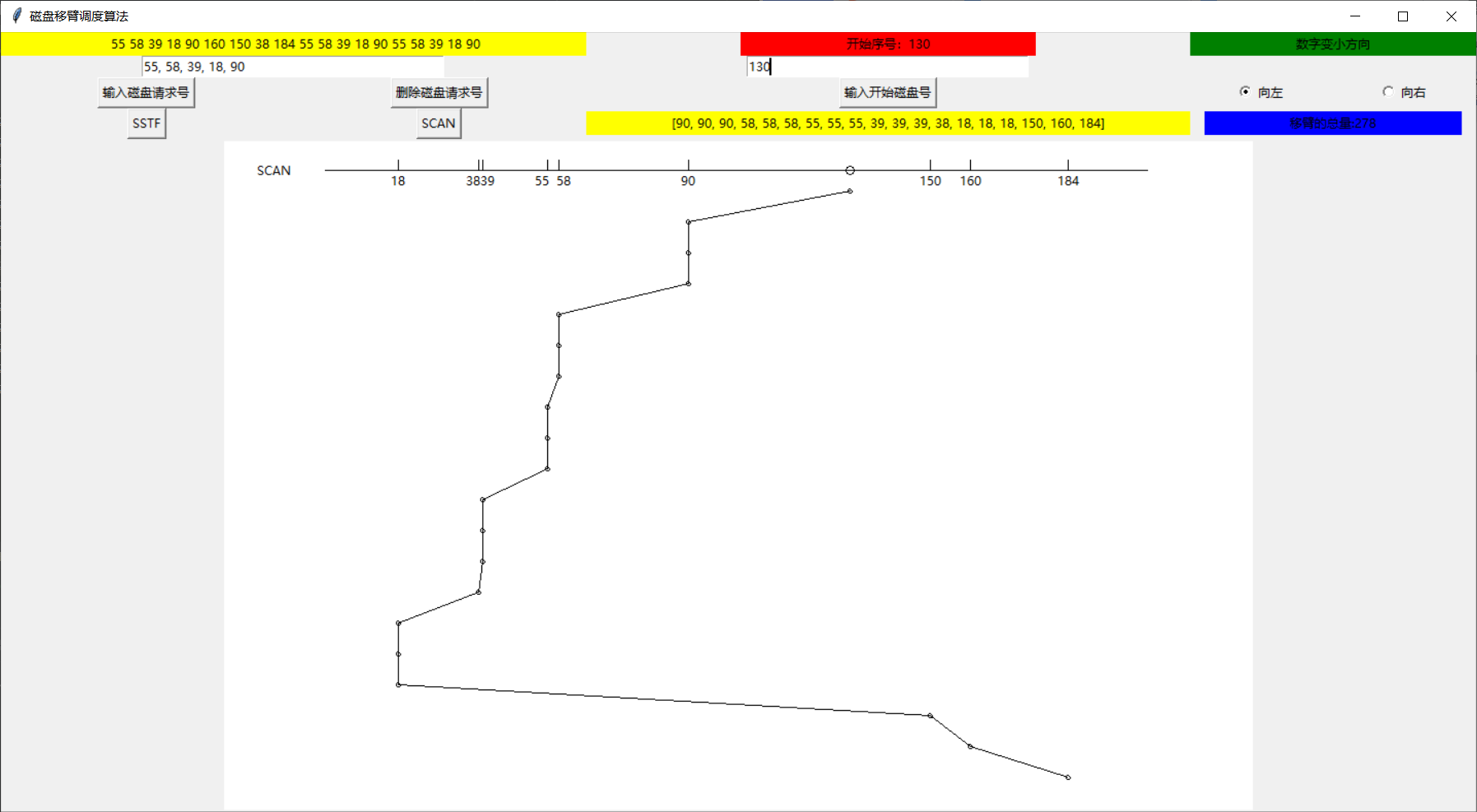


当请求磁盘号过长或开始磁盘号过大，自动调整比例



1. 结果展示





# 代码

（python3.7）

*from* tkinter *import* \*  
*import* tkinter.font *as* tf  
  
*def* is\_number(*s*):  
 *try*:  
 float(*s*)  
 *return True  
 except* ValueError:  
 *pass  
 try*:  
 *import* unicodedata  
 unicodedata.numeric(*s*)  
 *return True  
 except* (TypeError, ValueError):  
 *pass  
 return False  
  
def* GUI():  
 start = 0  
 disk\_num = []  
 direction = 0 # 0表示向数字减小的方向，1表示向数字增大的方向  
 record = []  
 ##整体窗口参数#################################################################  
 root\_window = Tk()  
 root\_window.title('磁盘移臂调度算法')  
 root\_window.resizable(width=*True*, height=*True*)  
 ft = tf.Font(size=6)  
 ##############################################################################  
  
 ##第一层三个标签###############################################################  
 disk\_label\_str = StringVar()  
 disk\_label\_str.set('磁盘请求序列：')  
 disk\_label = Label(root\_window, textvariable=disk\_label\_str, bg='yellow')  
 disk\_label.grid(row=0, column=0, columnspan=4, ipadx=100)  
 start\_label\_str = StringVar()  
 start\_label\_str.set('开始磁盘号：')  
 start\_label = Label(root\_window, textvariable=start\_label\_str, bg='red')  
 start\_label.grid(row=0, column=4, ipadx=100)  
 direction\_label\_str = StringVar()  
 direction\_label\_str.set('磁头移动方向：')  
 direction\_label = Label(root\_window, textvariable=direction\_label\_str, bg='green')  
 direction\_label.grid(row=0, column=5, ipadx=100, columnspan=2)  
 ##############################################################################  
  
 ##第二层三个输入框##############################################################  
 disk\_entry\_str = StringVar()  
 disk\_entry\_str.set('输入磁盘请求序列')  
 Entry(root\_window, textvariable=disk\_entry\_str).grid(row=1, column=0, columnspan=4, ipadx=75)  
 start\_entry\_str = StringVar()  
 start\_entry\_str.set('输入开始磁盘号')  
 Entry(root\_window, textvariable=start\_entry\_str).grid(row=1, column=4, ipadx=65)  
 # direction\_entry\_str = StringVar()  
 # direction\_entry\_str.set('输入磁头移动方向')  
 # Entry(root\_window, textvariable=direction\_entry\_str).grid(row=1, column=5, ipadx=70, columnspan=2)  
 ###############################################################################  
  
 ##第三层的四个按钮###############################################################  
 *def* get\_disk():  
 *nonlocal* disk\_num  
 new\_disk\_num = disk\_entry\_str.get()  
 *if* (is\_number(new\_disk\_num)):  
 disk\_num.append(int(new\_disk\_num))  
 *elif* (',' *in* new\_disk\_num):  
 new\_disk\_nums = new\_disk\_num.split(',')  
 *for* i *in* new\_disk\_nums:  
 disk\_num.append(int(i))  
 string = ''  
 *for* i *in* disk\_num:  
 string = string + ' ' + str(i)  
 disk\_label\_str.set(string)  
 *def* delete\_disk():  
 *nonlocal* disk\_num  
 *if* len(disk\_num) == 0:  
 *return  
 del* disk\_num[len(disk\_num) - 1]  
 *if* len(disk\_num) >= 1:  
 string = ''  
 *for* i *in* disk\_num:  
 string = string + ' ' + str(i)  
 disk\_label\_str.set(string)  
 *else*:  
 disk\_label\_str.set('输入磁盘请求序列')  
 *def* get\_start():  
 *nonlocal* start  
 new\_start = start\_entry\_str.get()  
 *if* (is\_number(new\_start)):  
 start = int(new\_start)  
 string = '开始序号：' + str(start)  
 start\_label\_str.set(string)  
 # def get\_direction():  
 # nonlocal direction  
 # new\_direction = direction\_entry\_str.get()  
 # if(is\_number(new\_direction)):  
 # if(int(new\_direction)==0 or int(new\_direction)==1):  
 # direction = int(new\_direction)  
 # if(int(direction) == 0):  
 # string = '数字变小方向'  
 # elif(int(direction) == 1):  
 # string = '数字变大方向'  
 # direction\_label\_str.set(string)  
 Button(root\_window, text='输入磁盘请求号', command=get\_disk).grid(row=2, column=0, columnspan=2)  
 Button(root\_window, text='删除磁盘请求号', command=delete\_disk).grid(row=2, column=2, columnspan=2)  
 Button(root\_window, text='输入开始磁盘号', command=get\_start).grid(row=2, column=4)  
 # Button(root\_window, text='输入磁头移动方向', command=get\_direction).grid(row=2, column=5)  
  
 *def* set\_direction\_left():  
 *nonlocal* direction  
 direction = v.get()  
 direction\_label\_str.set('数字变小方向')  
 *def* set\_direction\_right():  
 *nonlocal* direction  
 direction = v.get()  
 direction\_label\_str.set('数字变大方向')  
 v = IntVar()  
 Radiobutton(root\_window, text="向左", variable=v, value=0, command=set\_direction\_left).grid(row=2, column=5)  
 Radiobutton(root\_window, text="向右", variable=v, value=1, command=set\_direction\_right).grid(row=2, column=6)  
  
 ################################################################################  
  
 ##第五层帆布显示算法##############################################################  
 line\_start = 100  
 line\_end = 900  
 Height = 400 *if* len(disk\_num) \* 30 + 50 < 400 *else* len(disk\_num) \* 30 + 80  
 canvas = Canvas(root\_window, bg='white', confine=*False*)  
 canvas.config(width=1000, height=Height)  
 canvas.grid(row=4, column=0, columnspan=7, rowspan=2)  
  
 *def* create\_disk\_num\_line(*y*):  
 *nonlocal* disk\_num, line\_start, line\_end, start  
 *if* len(disk\_num) == 0:  
 *return* sorted\_disk\_num = sorted(list(set(disk\_num)))  
 canvas.create\_line(line\_start, *y*, line\_end, *y*)  
 *if* start <= max(sorted\_disk\_num) + 20:  
 start\_x = (line\_end-line\_start) \* (start / (max(sorted\_disk\_num) + 20)) + line\_start  
 canvas.create\_oval(start\_x-4, *y*-4, start\_x+4, *y*+4)  
 disk\_num\_x = []  
 *for* i *in* range(len(sorted\_disk\_num)):  
 disk\_num\_x.append((line\_end-line\_start) \* (sorted\_disk\_num[i] / (max(sorted\_disk\_num) + 20)) + line\_start)  
 str\_x = disk\_num\_x.copy()  
 *for* i *in* range(len(str\_x)):  
 *if* i != len(str\_x) - 1 *and* str\_x[i + 1] - str\_x[i] <= 15:  
 str\_x[i] -= 5  
 str\_x[i + 1] += 5  
 *for* i *in* range(len(str\_x)):  
 canvas.create\_line(disk\_num\_x[i], *y* - 10, disk\_num\_x[i], *y*)  
 canvas.create\_text(str\_x[i], *y* + 10, text=str(sorted\_disk\_num[i]))  
 *else*:  
 start\_x = (line\_end - line\_start) \* (start / (start + 20)) + line\_start  
 canvas.create\_oval(start\_x - 4, *y* - 4, start\_x + 4, *y* + 4)  
 disk\_num\_x = []  
 *for* i *in* range(len(sorted\_disk\_num)):  
 disk\_num\_x.append(  
 (line\_end - line\_start) \* (sorted\_disk\_num[i] / (start + 20)) + line\_start)  
 str\_x = disk\_num\_x.copy()  
 *for* i *in* range(len(str\_x)):  
 *if* i != len(str\_x) - 1 *and* str\_x[i + 1] - str\_x[i] <= 15:  
 str\_x[i] -= 5  
 str\_x[i + 1] += 5  
 *for* i *in* range(len(str\_x)):  
 canvas.create\_line(disk\_num\_x[i], *y* - 10, disk\_num\_x[i], *y*)  
 canvas.create\_text(str\_x[i], *y* + 10, text=str(sorted\_disk\_num[i]))  
 *def* create\_disk\_move(*y*):  
 *nonlocal* record, line\_start, line\_end, start  
 *if* len(record)==0:  
 *return  
 if* start <= max(record) + 20:  
 SSTF\_record\_x = [(line\_end-line\_start) \* (start / (max(record) + 20)) + line\_start]  
 *for* i *in* range(len(record)):  
 SSTF\_record\_x.append((line\_end-line\_start) \* (record[i] / (max(record) + 20)) + line\_start)  
 SSTF\_line = []  
 *for* i *in* range(len(SSTF\_record\_x)):  
 x = SSTF\_record\_x[i]  
 SSTF\_line.append([x, *y*])  
 canvas.create\_oval(x-2, *y*-2, x+2, *y*+2)  
 *y* += 30  
 *for* i *in* range(len(SSTF\_line)-1):  
 canvas.create\_line(tuple(SSTF\_line[i]), tuple(SSTF\_line[i+1]))  
 *else*:  
 SSTF\_record\_x = [(line\_end - line\_start) \* (start / (start + 20)) + line\_start]  
 *for* i *in* range(len(record)):  
 SSTF\_record\_x.append((line\_end - line\_start) \* (record[i] / (start + 20)) + line\_start)  
 SSTF\_line = []  
 *for* i *in* range(len(SSTF\_record\_x)):  
 x = SSTF\_record\_x[i]  
 SSTF\_line.append([x, *y*])  
 canvas.create\_oval(x - 2, *y* - 2, x + 2, *y* + 2)  
 *y* += 30  
 *for* i *in* range(len(SSTF\_line) - 1):  
 canvas.create\_line(tuple(SSTF\_line[i]), tuple(SSTF\_line[i + 1]))  
 ################################################################################  
  
 ##第四层两个算法按钮##############################################################  
 record\_label\_var = StringVar()  
 record\_label\_var.set('这里显示磁盘移动序列')  
 Label(root\_window, textvariable=record\_label\_var, bg='yellow').grid(row=3, column=4, ipadx=80)  
 move\_label\_var = StringVar()  
 move\_label\_var.set('这里显示移臂的总量')  
 Label(root\_window, textvariable=move\_label\_var, bg='blue').grid(row=3, column=5, columnspan=2, ipadx=80)  
  
 *def* show\_SSTF():  
 *nonlocal* record, start, disk\_num, line\_start  
 x = ALL  
 canvas.delete(x)  
 Height = 400 *if* len(disk\_num) \* 30 + 50 < 400 *else* len(disk\_num) \* 30 + 80  
 canvas.config(width=1000, height=Height)  
 canvas.create\_text(50, 30, text='SSTF')  
 record = SSTF(start, disk\_num.copy())  
 record\_label\_var.set(str(record))  
 last\_num = start  
 sum\_move = 0  
 *for* i *in* range(len(record)):  
 sum\_move += abs(record[i] - last\_num)  
 last\_num = record[i]  
 move\_label\_var.set('移臂的总量:' + str(sum\_move))  
 create\_disk\_num\_line(30)  
 create\_disk\_move(50)  
 *def* show\_SCAN():  
 *nonlocal* record, start, disk\_num, direction  
 x = ALL  
 canvas.delete(x)  
 Height = 400 *if* len(disk\_num) \* 30 + 50 < 400 *else* len(disk\_num) \* 30 + 80  
 canvas.config(width=1000, height=Height)  
 canvas.create\_text(50, 30, text='SCAN')  
 record = SCAN(start, disk\_num.copy(), direction)  
 record\_label\_var.set(str(record))  
 last\_num = start  
 sum\_move = 0  
 *for* i *in* range(len(record)):  
 sum\_move += abs(record[i] - last\_num)  
 last\_num = record[i]  
 move\_label\_var.set('移臂的总量:' + str(sum\_move))  
 create\_disk\_num\_line(30)  
 create\_disk\_move(50)  
 Button(root\_window, text='SSTF', command=show\_SSTF).grid(row=3, column=0, columnspan=2)  
 Button(root\_window, text='SCAN', command=show\_SCAN).grid(row=3, column=2, columnspan=2)  
 ################################################################################  
  
 root\_window.mainloop()  
*def* SSTF(*start*, *disk\_num*):  
 disk\_record = []  
 *for* i *in* range(len(*disk\_num*)):  
 next\_disk = sorted(*disk\_num*, key=*lambda x*:abs(x-*start*))[0]  
 disk\_record.append(next\_disk)  
 *disk\_num*.remove(next\_disk)  
 start = next\_disk  
 *return* disk\_record  
*def* SCAN(*start*, *disk\_num*, *direction*):  
 disk\_record = []  
 left\_num = []  
 right\_num = []  
 *for* i *in disk\_num*:  
 *if* i <= *start*:  
 left\_num.append(i)  
 *else*:  
 right\_num.append(i)  
 *if direction* == 0:  
 disk\_record = disk\_record + sorted(left\_num, reverse=*True*)  
 disk\_record = disk\_record + sorted(right\_num)  
 *elif direction* == 1:  
 disk\_record = disk\_record + sorted(right\_num)  
 disk\_record = disk\_record + sorted(left\_num, reverse=*True*)  
 *return* disk\_record  
  
  
  
  
  
*if* \_\_name\_\_ == '\_\_main\_\_':  
 # start = 100  
 # disk\_num = [55, 58, 39, 18, 90, 160, 150, 38, 184]  
 # direction = 0 # 0表示向数字减小的方向，1表示向数字增大的方向  
 # print(SSTF(start, disk\_num.copy()))  
 # print(SCAN(start, disk\_num.copy(), 0))  
 # print(SCAN(start, disk\_num.copy(), 1))  
  
 GUI()