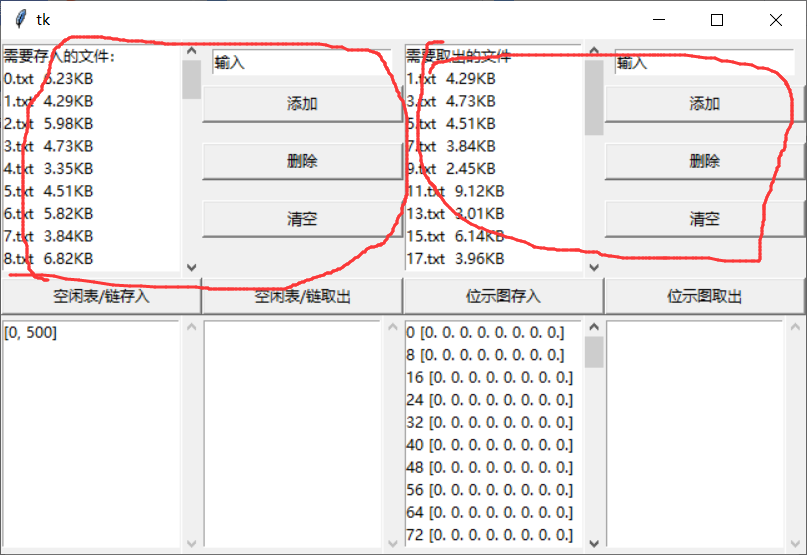
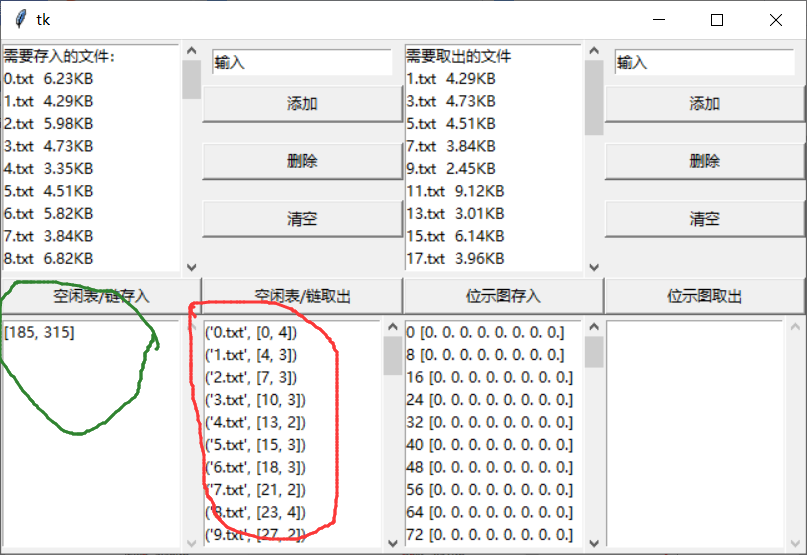
# 实验结果

1. 初始化界面



在左上角显示要存入的文件名和文件大小，旁边可以输入（形如1.txt,12，单位默认为KB），删除（删除最新添加的），清空

在右上角显示要取出的文件名和文件大小，旁边可以输入（形如1.txt,12，单位默认为KB），删除（删除最新添加的），清空

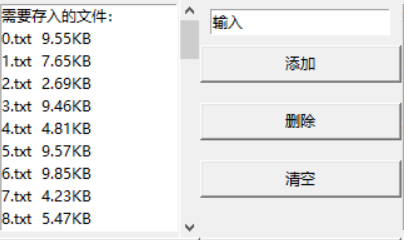


左下角显示空闲表/空闲盘区链（因为我使用的是python，Python对指针已经做了良性封装，任何对象都可以被引用，所以Python一般并不需要指针或者链表，因而我也没有单独去编写链表结构，而是统一使用Python自带的list结构存储），显示是形式如图（图中绿色）所示：[0, 500]表示第一块空闲块块号为0，有连续500个空闲块。图中红色表示文件存储状态：如(‘0.txt’, [0, 4])表示文件0.txt存储在第0块，用了包括第0块在内的4个块（即0,1,2,3块）。

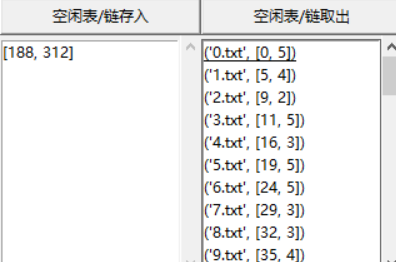
位示图的显示同空闲表类似。

1. 存入50个2~10KB的文件

50个文件初始已经创建好了



点击空闲表/链存入，存入空闲表：



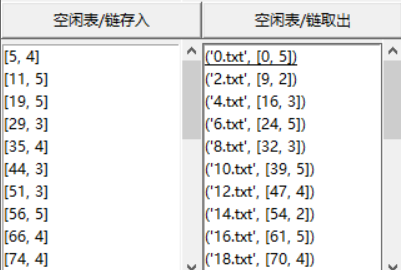
点击位示图存入：



1. 删除奇数文件

需要删除的奇数文件也已经在初始化时创建好了

点击空闲表/链取出：

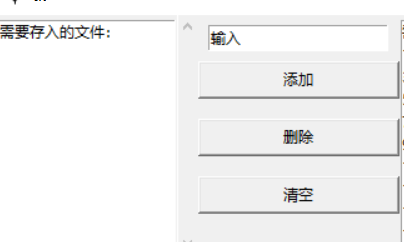


点击位示图取出

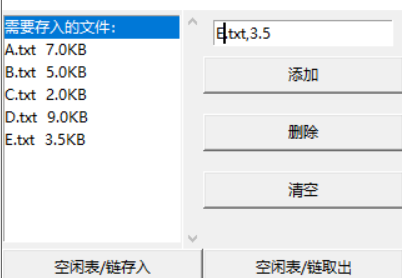


1. 新建文件并存入

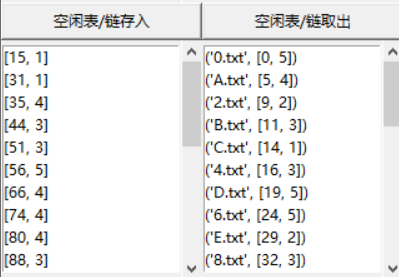
点击清空，清空需要存入的文件：



分别输入A.txt、B.txt、C.txt、D.txt、E.txt，大小为：7k、5k、2k、9k、3.5k，



存入空闲表/链：



存入位示图：



（5）给出文件A.txt、B.txt、C.txt、D.txt、E.txt的盘块存储状态和所有空闲区块的状态。

空闲表：

[[22, 1],

[28, 4],

[36, 2],

[40, 3],

[46, 2],

[51, 3],

[58, 4],

[66, 4],

[74, 3],

[79, 3],

[85, 4],

[93, 3],

[98, 4],

[104, 4],

[120, 5],

[127, 3],

[135, 5],

[142, 5],

[151, 2],

[156, 3],

[162, 5],

[171, 329]]

文件存储状态：

('A.txt', [4, 4]),

('B.txt', [11, 3]),

('C.txt', [14, 1]),

('D.txt', [113, 5]),

('E.txt', [20, 2])

空闲盘区链：

[[22, 1],

[28, 4],

[36, 2],

[40, 3],

[46, 2],

[51, 3],

[58, 4],

[66, 4],

[74, 3],

[79, 3],

[85, 4],

[93, 3],

[98, 4],

[104, 4],

[120, 5],

[127, 3],

[135, 5],

[142, 5],

[151, 2],

[156, 3],

[162, 5],

[171, 329]]

文件存储状态：

('A.txt', [4, 4]),

('B.txt', [11, 3]),

('C.txt', [14, 1]),

('D.txt', [113, 5]),

('E.txt', [20, 2])

位示图：

array([[1., 1., 1., 1., 1., 1., 1., 1.],

[1., 1., 1., 1., 1., 1., 1., 1.],

[1., 1., 1., 1., 1., 1., 0., 1.],

[1., 1., 1., 1., 0., 0., 0., 0.],

[1., 1., 1., 1., 0., 0., 1., 1.],

[0., 0., 0., 1., 1., 1., 0., 0.],

[1., 1., 1., 0., 0., 0., 1., 1.],

[1., 1., 0., 0., 0., 0., 1., 1.],

[1., 1., 0., 0., 0., 0., 1., 1.],

[1., 1., 0., 0., 0., 1., 1., 0.],

[0., 0., 1., 1., 1., 0., 0., 0.],

[0., 1., 1., 1., 1., 0., 0., 0.],

[1., 1., 0., 0., 0., 0., 1., 1.],

[0., 0., 0., 0., 1., 1., 1., 1.],

[1., 1., 1., 1., 1., 1., 1., 1.],

[0., 0., 0., 0., 0., 1., 1., 0.],

[0., 0., 1., 1., 1., 1., 1., 0.],

[0., 0., 0., 0., 1., 1., 0., 0.],

[0., 0., 0., 1., 1., 1., 1., 0.],

[0., 1., 1., 1., 0., 0., 0., 1.],

[1., 1., 0., 0., 0., 0., 0., 1.],

[1., 1., 1., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.],

[0., 0., 0., 0., 0., 0., 0., 0.]])

文件存储状态：

('A.txt', [4, 4]),

('B.txt', [11, 3]),

('C.txt', [14, 1]),

('D.txt', [113, 5]),

('E.txt', [20, 2])

# 代码

*import* numpy *as* np  
*from* tkinter *import* \*  
*from* pprint *import* pprint  
# 参数  
KB = 2048  
size\_of\_disk\_block = 2\*KB  
num\_of\_disk\_block = 500  
num\_of\_file = 50  
# 结构  
free\_table = [[0, num\_of\_disk\_block]]  
bitmap = np.zeros(num\_of\_disk\_block)  
# 磁盘  
disk = np.zeros((num\_of\_disk\_block, size\_of\_disk\_block))  
#########################################################################################  
*def* GUI(*num\_file\_dict*, *delete\_file\_dict*):  
 *global* free\_table, bitmap, KB, num\_of\_file, num\_of\_disk\_block  
 free\_table\_file\_table = {}  
 bitmap\_file\_table = {}  
 add\_file\_dict = *num\_file\_dict* delete\_file\_dict = *delete\_file\_dict* input\_scrollbar\_row = 0  
 button\_row = 1  
 show\_scrollbar\_row = 6  
  
 root\_window = Tk()  
 #################################################################################  
 # 第一层两个滑动窗口  
 add\_file\_scrollbar = Scrollbar(root\_window)  
 add\_file\_scrollbar.grid(row=input\_scrollbar\_row, column=1, sticky=N + E + W, ipady=70, rowspan=5)  
 add\_file\_listbox = Listbox(root\_window, yscrollcommand=add\_file\_scrollbar.set)  
 add\_file\_listbox.insert(END, '需要存入的文件：')  
 *for* file\_name, file\_text *in* add\_file\_dict.items():  
 file\_size = np.round(len(file\_text) / KB, 2)  
 add\_file\_listbox.insert(END, file\_name + ' ' + str(file\_size) + 'KB')  
 add\_file\_listbox.grid(row=input\_scrollbar\_row, column=0, rowspan=5)  
 ###############  
 delete\_file\_scrollbar = Scrollbar(root\_window)  
 delete\_file\_scrollbar.grid(row=input\_scrollbar\_row, column=5, sticky=N + E + W, ipady=70, rowspan=5)  
 delete\_file\_listbox = Listbox(root\_window, yscrollcommand=delete\_file\_scrollbar.set)  
 delete\_file\_listbox.insert(END, '需要取出的文件')  
 *for* file\_name, file\_text *in delete\_file\_dict*.items():  
 file\_size = np.round(len(file\_text) / KB, 2)  
 delete\_file\_listbox.insert(END, file\_name + ' ' + str(file\_size) + 'KB')  
 delete\_file\_listbox.grid(row=input\_scrollbar\_row, column=4, rowspan=5)  
 ###########################################################################################  
  
  
 #########################################################################################################  
 # 左边四个  
 input\_add\_file\_entry\_str = StringVar()  
 input\_add\_file\_entry\_str.set('输入')  
 Entry(root\_window, textvariable=input\_add\_file\_entry\_str).grid(row=0, column=2, columnspan=2)  
  
 *def* get\_add\_file():  
 *nonlocal* add\_file\_dict  
 new\_file = input\_add\_file\_entry\_str.get()  
 file\_name = new\_file.split(',')[0]  
 file\_size = float(new\_file.split(',')[1])  
 file\_text = np.random.randint(0, 2, size=int(file\_size\*KB))  
 add\_file\_dict[file\_name] = file\_text  
 add\_file\_listbox.insert(END, file\_name + ' ' + str(round(file\_size, 2)) + 'KB')  
 *def* delete\_add\_file():  
 all\_key = list(add\_file\_dict.keys())  
 # print(all\_key[1])  
 add\_file\_dict.pop(all\_key[-1])  
 add\_file\_listbox.delete(END)  
 *def* clear\_add\_file():  
 add\_file\_listbox.delete(0,END)  
 add\_file\_dict.clear()  
 add\_file\_listbox.insert(END, '需要存入的文件：')  
  
 Button(root\_window, text='添加', command=get\_add\_file).grid(row=button\_row, column=2, sticky=N + E + W, columnspan=2)  
 Button(root\_window, text='删除', command=delete\_add\_file).grid(row=button\_row+1, column=2, sticky=N + E + W, columnspan=2)  
 Button(root\_window, text='清空', command=clear\_add\_file).grid(row=button\_row+2, column=2, sticky=N + E + W, columnspan=2)  
 ################################################################################################################  
  
 #########################################################################################################  
 # 右边4个  
 input\_delete\_file\_entry\_str = StringVar()  
 input\_delete\_file\_entry\_str.set('输入')  
 Entry(root\_window, textvariable=input\_delete\_file\_entry\_str).grid(row=0, column=6, columnspan=2)  
  
 *def* get\_delete\_file():  
 *nonlocal* delete\_file\_dict  
 new\_file = input\_add\_file\_entry\_str.get()  
 file\_name = new\_file.split(',')[0]  
 file\_size = float(new\_file.split(',')[1])  
 file\_text = np.random.randint(0, 2, size=int(file\_size \* KB))  
 delete\_file\_dict[file\_name] = file\_text  
 delete\_file\_listbox.insert(END, file\_name + ' ' + str(round(file\_size, 2)) + 'KB')  
 *def* delete\_delete\_file():  
 all\_key = list(delete\_file\_dict.keys())  
 # print(all\_key[1])  
 delete\_file\_dict.pop(all\_key[-1])  
 delete\_file\_listbox.delete(END)  
 *def* clear\_delete\_file():  
 delete\_file\_listbox.delete(0,END)  
 delete\_file\_dict.clear()  
 delete\_file\_listbox.insert(END, '需要取出的文件：')  
  
 Button(root\_window, text='添加', command=get\_delete\_file).grid(row=button\_row, column=6, sticky=N + E + W, columnspan=2)  
 Button(root\_window, text='删除', command=delete\_delete\_file).grid(row=button\_row + 1, column=6, sticky=N + E + W,  
 columnspan=2)  
 Button(root\_window, text='清空', command=clear\_delete\_file).grid(row=button\_row + 2, column=6, sticky=N + E + W,  
 columnspan=2)  
 ###############################################################################################################  
  
  
 # free\_table####################################################################  
 # 空闲表  
  
 free\_table\_scrollbar = Scrollbar(root\_window)  
 free\_table\_scrollbar.grid(row=show\_scrollbar\_row, column=1, sticky=N + E + W, ipady=70)  
 free\_table\_listbox = Listbox(root\_window, yscrollcommand=free\_table\_scrollbar.set)  
 *for* i *in* range(len(free\_table)):  
 row = free\_table[i]  
 free\_table\_listbox.insert(END, str(row))  
 free\_table\_listbox.grid(row=show\_scrollbar\_row, column=0)  
 # file\_table####################################################################  
 free\_table\_file\_table\_scrollbar = Scrollbar(root\_window)  
 free\_table\_file\_table\_scrollbar.grid(row=show\_scrollbar\_row, column=3, sticky=N + E + W, ipady=70)  
 free\_table\_file\_table\_listbox = Listbox(root\_window, yscrollcommand=free\_table\_file\_table\_scrollbar.set)  
 new\_free\_table\_file\_table = sorted(free\_table\_file\_table.items(), key=*lambda x*:x[1][0])  
 *for* row *in* new\_free\_table\_file\_table:  
 free\_table\_file\_table\_listbox.insert(END, str(row))  
 free\_table\_file\_table\_listbox.grid(row=show\_scrollbar\_row, column=2)  
 free\_table\_file\_table\_scrollbar.config(command=free\_table\_file\_table\_listbox.yview)  
 ############################################################################  
  
  
 # bitmap####################################################################  
  
 add\_num\_of\_bitmap = 8 - (num\_of\_disk\_block % 8)  
 add\_zeros = np.zeros(add\_num\_of\_bitmap)  
 new\_bitmap = np.hstack((bitmap, add\_zeros))  
 new\_bitmap = np.reshape(new\_bitmap, (len(new\_bitmap) // 8, 8))  
  
 bitmap\_scrollbar = Scrollbar(root\_window)  
 bitmap\_scrollbar.grid(row=show\_scrollbar\_row, column=5, sticky=N + E + W, ipady=70)  
 bitmap\_listbox = Listbox(root\_window, yscrollcommand=bitmap\_scrollbar.set)  
 *for* i *in* range(len(new\_bitmap)):  
 row = new\_bitmap[i]  
 bitmap\_listbox.insert(END, str(i \* 8) + ' ' + str(row))  
 bitmap\_listbox.grid(row=show\_scrollbar\_row, column=4)  
 # file\_table####################################################################  
 bitmap\_file\_table\_scrollbar = Scrollbar(root\_window)  
 bitmap\_file\_table\_scrollbar.grid(row=show\_scrollbar\_row, column=7, sticky=N + E + W, ipady=70)  
 bitmap\_file\_table\_listbox = Listbox(root\_window, yscrollcommand=bitmap\_file\_table\_scrollbar.set)  
 new\_bitmap\_file\_table = sorted(bitmap\_file\_table.items(), key=*lambda x*: x[1][0])  
 *for* row *in* new\_bitmap\_file\_table:  
 bitmap\_file\_table\_listbox.insert(END, str(row) + ' ' + str())  
 bitmap\_file\_table\_listbox.grid(row=show\_scrollbar\_row, column=6)  
 bitmap\_file\_table\_scrollbar.config(command=bitmap\_file\_table\_listbox.yview)  
 ###################################################################################  
  
 *def* free\_table\_add():  
 *for* file\_name, file\_text *in* add\_file\_dict.items():  
 AddFile2FreeTable(file\_name, file\_text, free\_table\_file\_table)  
  
 free\_table\_listbox.delete(0, END)  
 *for* i *in* range(len(free\_table)):  
 row = free\_table[i]  
 free\_table\_listbox.insert(END, str(row))  
  
 free\_table\_file\_table\_listbox.delete(0, END)  
 new\_free\_table\_file\_table = sorted(free\_table\_file\_table.items(), key=*lambda x*: x[1][0])  
 *for* row *in* new\_free\_table\_file\_table:  
 free\_table\_file\_table\_listbox.insert(END, str(row))  
  
 *def* free\_table\_delete():  
 *for* file\_name, file\_size *in* delete\_file\_dict.items():  
 DeleteFileInFreeTable(file\_name, free\_table\_file\_table)  
  
 free\_table\_listbox.delete(0, END)  
 *for* i *in* range(len(free\_table)):  
 row = free\_table[i]  
 free\_table\_listbox.insert(END, str(row))  
  
 free\_table\_file\_table\_listbox.delete(0, END)  
 new\_free\_table\_file\_table = sorted(free\_table\_file\_table.items(), key=*lambda x*: x[1][0])  
 *for* row *in* new\_free\_table\_file\_table:  
 free\_table\_file\_table\_listbox.insert(END, str(row))  
  
 *def* bitmap\_add():  
 *for* file\_name, file\_text *in* add\_file\_dict.items():  
 AddFile2BitMap(file\_name, file\_text, bitmap\_file\_table)  
  
 add\_num\_of\_bitmap = 8 - (num\_of\_disk\_block % 8)  
 add\_zeros = np.zeros(add\_num\_of\_bitmap)  
 new\_bitmap = np.hstack((bitmap, add\_zeros))  
 new\_bitmap = np.reshape(new\_bitmap, (len(new\_bitmap) // 8, 8))  
 bitmap\_listbox.delete(0, END)  
 *for* i *in* range(len(new\_bitmap)):  
 row = new\_bitmap[i]  
 bitmap\_listbox.insert(END, str(i \* 8) + ' ' + str(row))  
  
 new\_bitmap\_file\_table = sorted(bitmap\_file\_table.items(), key=*lambda x*: x[1][0])  
 bitmap\_file\_table\_listbox.delete(0, END)  
 *for* row *in* new\_bitmap\_file\_table:  
 bitmap\_file\_table\_listbox.insert(END, str(row) + ' ' + str())  
  
 *def* bitmap\_delete():  
 *for* file\_name, file\_size *in* delete\_file\_dict.items():  
 DeleteFileInBitMap(file\_name, bitmap\_file\_table)  
  
 add\_num\_of\_bitmap = 8 - (num\_of\_disk\_block % 8)  
 add\_zeros = np.zeros(add\_num\_of\_bitmap)  
 new\_bitmap = np.hstack((bitmap, add\_zeros))  
 new\_bitmap = np.reshape(new\_bitmap, (len(new\_bitmap) // 8, 8))  
 bitmap\_listbox.delete(0, END)  
 *for* i *in* range(len(new\_bitmap)):  
 row = new\_bitmap[i]  
 bitmap\_listbox.insert(END, str(i \* 8) + ' ' + str(row))  
  
 new\_bitmap\_file\_table = sorted(bitmap\_file\_table.items(), key=*lambda x*: x[1][0])  
 # print(len(bitmap\_file\_table))  
 bitmap\_file\_table\_listbox.delete(0, END)  
 *for* row *in* new\_bitmap\_file\_table:  
 bitmap\_file\_table\_listbox.insert(END, str(row) + ' ' + str())  
  
 Button(root\_window, text='空闲表/链存入', command=free\_table\_add).grid(row=5, column=0, sticky=N + E + W, columnspan=2)  
 Button(root\_window, text='空闲表/链取出', command=free\_table\_delete).grid(row=5, column=2, sticky=N + E + W,  
 columnspan=2)  
 Button(root\_window, text='位示图存入', command=bitmap\_add).grid(row=5, column=4, sticky=N + E + W, columnspan=2)  
 Button(root\_window, text='位示图取出', command=bitmap\_delete).grid(row=5, column=6, sticky=N + E + W,  
 columnspan=2)  
  
  
   
  
 root\_window.mainloop()  
#########################################################################################  
*def* AddFile2FreeTable(*file\_name*, *file\_text*, *file\_table*):  
 *global* KB, size\_of\_disk\_block, num\_of\_disk\_block, free\_table, disk  
 *if* len(*file\_text*) % size\_of\_disk\_block == 0:  
 file\_size = len(*file\_text*) // size\_of\_disk\_block  
 new\_file\_text = *file\_text  
 else*:  
 raw\_file\_size = len(*file\_text*) // size\_of\_disk\_block  
 file\_size = raw\_file\_size + 1  
 add\_text\_size = file\_size \* size\_of\_disk\_block - len(*file\_text*)  
 add\_text = np.ones(add\_text\_size) \* -1  
 new\_file\_text = np.hstack((*file\_text*, add\_text))  
 can\_save = 0  
 *for* i *in* range(len(free\_table)):  
 *if* free\_table[i][1] >= file\_size:  
 can\_save = 1  
 disk\_start\_num = free\_table[i][0]  
 free\_table[i][0] += file\_size  
 free\_table[i][1] -= file\_size  
 # print(':', disk\_start\_num, disk\_start\_num + file\_size)  
 # file\_table.append([file\_name, disk\_start\_num, file\_size])  
 *file\_table*[*file\_name*] = [disk\_start\_num, file\_size]  
 *for* j *in* range(disk\_start\_num, disk\_start\_num + file\_size):  
 # print('j:', j)  
 # print('start:', (j - disk\_start\_num) \* size\_of\_disk\_block)  
 # print('end:', ((j - disk\_start\_num) + 1) \* size\_of\_disk\_block)  
 # print('len:', len(new\_file\_text))  
 disk[j] = new\_file\_text[(j - disk\_start\_num) \* size\_of\_disk\_block:((j - disk\_start\_num) + 1) \* size\_of\_disk\_block]  
 *break  
 if* can\_save == 0:  
 # print(free\_table[i], file\_size)  
 # pprint(free\_table)  
 # pprint(file\_table)  
 print(*file\_name*, ' can not find enough block ', file\_size)  
 *return* new\_free\_table = []  
 *for* node *in* free\_table:  
 *if* node[1] != 0:  
 new\_free\_table.append(node)  
 free\_table = new\_free\_table  
  
  
*def* DeleteFileInFreeTable(*file\_name*, *file\_table*):  
 *global* KB, size\_of\_disk\_block, num\_of\_disk\_block, free\_table, disk  
 # pprint(free\_table)  
 *if file\_name not in file\_table*.keys():  
 # print(file\_name)  
 print('can not find ', *file\_name*)  
 *return* free\_table.append(*file\_table*[*file\_name*])  
 free\_table = sorted(free\_table)  
 new\_free\_table = []  
 *for* i *in* range(len(free\_table) - 1):  
 # pprint('\*')  
 # print(i)  
 # pprint(free\_table)  
 *if* free\_table[i][0] + free\_table[i][1] == free\_table[i + 1][0]:  
 free\_table[i][1] = free\_table[i][1] + free\_table[i + 1][1]  
 free\_table[i + 1][0] = -1  
 # free\_table[i][0] = -1  
 # free\_table.append([free\_table[i][0], free\_table[i][1] + free\_table[i + 1][1]])  
 *for* i *in* range(len(free\_table)):  
 *if* free\_table[i][0] != -1:  
 new\_free\_table.append(free\_table[i])  
 free\_table = new\_free\_table  
 *del file\_table*[*file\_name*]  
  
  
*def* AddFile2BitMap(*file\_name*, *file\_text*, *file\_table*):  
 *global* KB, size\_of\_disk\_block, num\_of\_disk\_block, bitmap, disk  
 *if* len(*file\_text*) % size\_of\_disk\_block == 0:  
 file\_size = len(*file\_text*) // size\_of\_disk\_block  
 new\_file\_text = *file\_text  
 else*:  
 raw\_file\_size = len(*file\_text*) // size\_of\_disk\_block  
 file\_size = raw\_file\_size + 1  
 add\_text\_size = file\_size \* size\_of\_disk\_block - len(*file\_text*)  
 add\_text = np.ones(add\_text\_size) \* -1  
 new\_file\_text = np.hstack((*file\_text*, add\_text))  
 can\_save = 0  
 num\_of\_zero = 0  
 start\_of\_zero = 0  
 *for* i *in* range(num\_of\_disk\_block):  
 *if* bitmap[i] == 0:  
 num\_of\_zero += 1  
 *else*:  
 num\_of\_zero = 0  
 start\_of\_zero = i + 1  
 *if* num\_of\_zero >= file\_size:  
 can\_save = 1  
 bitmap[start\_of\_zero:start\_of\_zero + num\_of\_zero] = np.ones(num\_of\_zero)  
 *file\_table*[*file\_name*] = [start\_of\_zero, num\_of\_zero]  
 *for* j *in* range(start\_of\_zero, start\_of\_zero + num\_of\_zero):  
 disk[j] = new\_file\_text[(j - start\_of\_zero) \* size\_of\_disk\_block:((j - start\_of\_zero) + 1) \* size\_of\_disk\_block]  
 *break  
 if* can\_save == 0:  
 print(*file\_name*, ' can not find enough block ', file\_size)  
 *return  
 return  
  
  
def* DeleteFileInBitMap(*file\_name*, *file\_table*):  
 *global* KB, size\_of\_disk\_block, num\_of\_disk\_block, bitmap, disk  
 *if file\_name not in file\_table*.keys():  
 # print(file\_name)  
 print('can not find ', *file\_name*)  
 *return* file\_node = *file\_table*[*file\_name*]  
 bitmap[file\_node[0]:file\_node[0] + file\_node[1]] = np.zeros(file\_node[1])  
 *del file\_table*[*file\_name*]  
  
  
  
*if* \_\_name\_\_ == '\_\_main\_\_':  
  
 # 创建50个新文件  
 file\_table = {}  
 num\_file\_dict = {}  
 delete\_file\_dict = {}  
 *for* i *in* range(num\_of\_file):  
 file\_size = np.random.randint(2\*KB, 10\*KB, size=1)  
 file\_text = np.random.randint(0, 2, size=file\_size)  
 num\_file\_dict[str(i) + '.txt'] = file\_text  
 *if* i % 2 != 0:  
 delete\_file\_dict[str(i) + '.txt'] = file\_text  
 # 创建5个新文件  
 alpha\_file\_dict = {}  
 *for* file\_name, file\_size *in* zip(['A', 'B', 'C', 'D', 'E'], [7 \* KB, 5 \* KB, 2 \* KB, 9 \* KB, 3.5 \* KB]):  
 file\_text = np.random.randint(0, 2, size=int(file\_size))  
 alpha\_file\_dict[file\_name + '.txt'] = file\_text  
  
 # # 空闲表/空闲盘区链测试  
 # for file\_name, file\_text in num\_file\_dict.items():  
 # AddFile2FreeTable(file\_name, file\_text, file\_table)  
 # # print('-'\*30)  
 # # pprint(free\_table)  
 # # pprint(sorted(file\_table.items(), key=lambda x: x[1]))  
 # for i in range(1, num\_of\_file, 2):  
 # DeleteFileInFreeTable(str(i) + '.txt', file\_table)  
 # # print('-' \* 30)  
 # # pprint(free\_table)  
 # # pprint(sorted(file\_table.items(), key=lambda x: x[1]))  
 # for file\_name, file\_text in alpha\_file\_dict.items():  
 # AddFile2FreeTable(file\_name, file\_text, file\_table)  
 # print('-' \* 30)  
 # pprint(free\_table)  
 # pprint(sorted(file\_table.items(), key=lambda x:x[1]))  
 #   
 # # 位示图测试  
 # file\_table.clear()  
 # for file\_name, file\_text in num\_file\_dict.items():  
 # AddFile2BitMap(file\_name, file\_text, file\_table)  
 # for i in range(1, num\_of\_file, 2):  
 # DeleteFileInBitMap(str(i) + '.txt', file\_table)  
 # for file\_name, file\_text in alpha\_file\_dict.items():  
 # AddFile2BitMap(file\_name, file\_text, file\_table)  
 # add\_num\_of\_bitmap = 8 - (num\_of\_disk\_block % 8)  
 # add\_zeros = np.zeros(add\_num\_of\_bitmap)  
 # new\_bitmap = np.hstack((bitmap, add\_zeros))  
 # pprint(np.reshape(new\_bitmap, (len(new\_bitmap) // 8, 8)))  
 # pprint(sorted(file\_table.items(), key=lambda x:x[1]))  
  
  
 # # 界面  
 GUI(num\_file\_dict, delete\_file\_dict)