目录

[实验结果展示 2](#_Toc3616)

[代码 6](#_Toc7643)

# 实验结果展示

#### 初始界面



#### 输入

（1）输入1，点击输入页号 （2）输入2，点击输入页号

（3）点击删除页号 （4）点击删除页号

（5）输入除了数字的其他字符 （6）单个输入但费劲，支持连续输入，形式 如：1,2,3,4（逗号为英文逗号，中文逗 号会在第一次自动转换成英文逗号）

 （7）输入内存大小4，点击输入内存大小 （8）输入其他字符，点击输入内存大小， 并不会改变内存大小

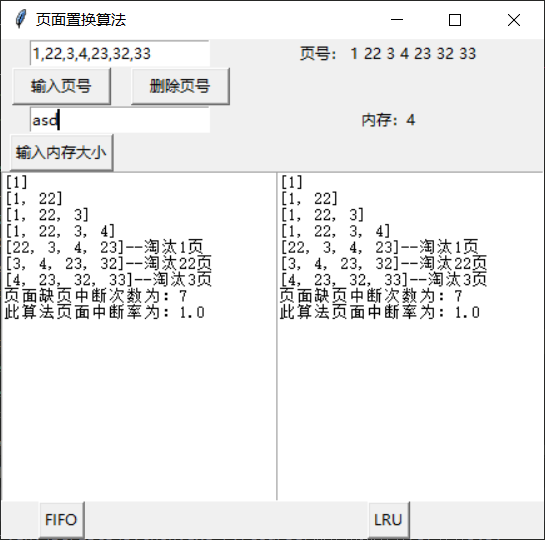
 

#### 运行算法

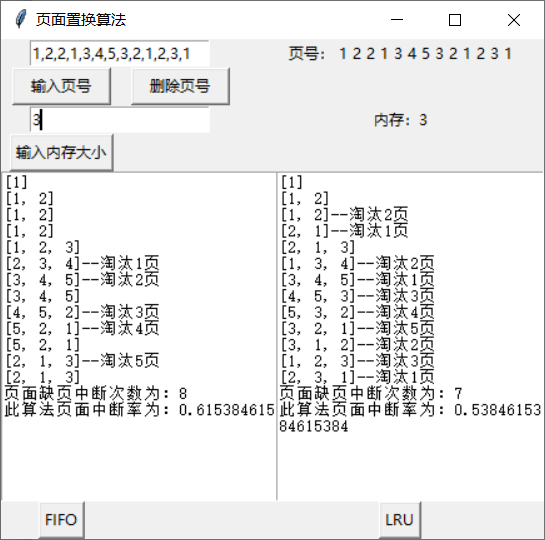
1. 点击FIFO



1. 点击LRU



1. 换一组数据



# 代码

（python3.7）

*from* tkinter *import* \*  
  
*def* GUI():  
 pages = []  
 memory\_size = 0  
 ##整体窗口参数#################################################################  
 root\_window = Tk()  
 root\_window.title('页面置换算法')  
 # root\_window.geometry('200x100')  
 root\_window.resizable(width=*True*, height=*True*)  
 ##############################################################################  
  
 ##子窗口1#####################################################################  
 # 页号输入框  
 page\_num\_entry = StringVar()  
 page\_num\_entry.set('请输入页号')  
 Entry(root\_window, textvariable=page\_num\_entry).grid(row=0, column=0, columnspan=2)  
 # 内存大小输入框  
 memory\_size\_entry = StringVar()  
 memory\_size\_entry.set('请输入内存大小')  
 Entry(root\_window, textvariable=memory\_size\_entry).grid(row=2, column=0, columnspan=2)  
 # 显示页号  
 show\_pages\_label\_text = StringVar()  
 show\_pages\_label\_text.set('在这里显示页号')  
 show\_pages\_label = Label(root\_window, textvariable=show\_pages\_label\_text)  
 show\_memory\_size\_label\_text = StringVar()  
 show\_memory\_size\_label\_text.set('在这里显示内存大小')  
 show\_memory\_size\_label = Label(root\_window, textvariable=show\_memory\_size\_label\_text)  
 *def* get\_page():  
 *nonlocal* pages  
 page\_num = page\_num\_entry.get()  
 *if*(is\_number(page\_num)):  
 pages.append(int(page\_num))  
 *elif*(',' *in* page\_num):  
 page\_nums = page\_num.split(',')  
 *for* i *in* page\_nums:  
 pages.append(int(i))  
 string = '页号：'  
 *for* i *in* pages:  
 string = string + ' ' + str(i)  
 show\_pages\_label\_text.set(string)  
 *def* delete\_page():  
 *nonlocal* pages  
 *if* len(pages) == 0:  
 *return  
 del* pages[len(pages) - 1]  
 *if* len(pages) >= 1:  
 string = '页号：'  
 *for* i *in* pages:  
 string = string + ' ' + str(i)  
 show\_pages\_label\_text.set(string)  
 *else*:  
 show\_pages\_label\_text.set('在这里显示页号')  
 *def* get\_memory\_size():  
 *nonlocal* memory\_size  
 memory\_size\_num = memory\_size\_entry.get()  
 *if* (is\_number(memory\_size\_num)):  
 memory\_size = int(memory\_size\_num)  
 string = '内存：' + str(memory\_size)  
 show\_memory\_size\_label\_text.set(string)  
 # print(memory\_size)  
 Button(root\_window, text='输入页号', command=get\_page).grid(row=1, column=0, ipadx=10)  
 Button(root\_window, text='删除页号', command=delete\_page).grid(row=1, column=1, ipadx=10)  
 Button(root\_window, text='输入内存大小', command=get\_memory\_size).grid(row=3, column=0)  
 show\_pages\_label.grid(row=0, column=4)  
 show\_memory\_size\_label.grid(row=2, column=4)  
 ###################################################################################  
  
 ##子窗口2###########################################################################  
 *def* show\_FIFO():  
 FIFO\_text.delete(1.0, END)  
 FIFO\_delete, FIFO\_page, error\_num, rate = FIFO(pages, memory\_size)  
 *for* i *in* range(len(FIFO\_page)):  
 text = FIFO\_page[i]  
 *if* FIFO\_delete[i] == 0:  
 FIFO\_text.insert(END, str(text) + '\n')  
 *else*:  
 FIFO\_text.insert(END, str(text) + '--淘汰' + str(FIFO\_delete[i]) + '页\n')  
 FIFO\_text.insert(END, '页面缺页中断次数为：' + str(error\_num) + '\n')  
 FIFO\_text.insert(END, '此算法页面中断率为：' + str(rate))  
 *def* show\_LRU():  
 LRU\_text.delete(1.0, END)  
 LRU\_delete, LRU\_page, error\_num, rate = LRU(pages, memory\_size)  
 *for* i *in* range(len(LRU\_page)):  
 text = LRU\_page[i]  
 *if* LRU\_delete[i] == 0:  
 LRU\_text.insert(END, str(text) + '\n')  
 *else*:  
 LRU\_text.insert(END, str(text) + '--淘汰' + str(LRU\_delete[i]) + '页\n')  
 LRU\_text.insert(END, '页面缺页中断次数为：' + str(error\_num) + '\n')  
 LRU\_text.insert(END, '此算法页面中断率为：' + str(rate))  
 # 子窗口配置（包含FIFO子窗口和LRU子窗口）  
 frame2 = Frame(root\_window)  
 FIFO\_frame = Frame(frame2)  
 FIFO\_text = Text(root\_window, width=30, height=20)  
 FIFO\_text.grid(row=4, column=0, rowspan=3, columnspan=6, ipadx=110)  
 Button(root\_window, text='FIFO', command=show\_FIFO).grid(row=10, column=0)  
 LRU\_frame = Frame(frame2)  
 LRU\_text = Text(root\_window, width=30, height=20)  
 LRU\_text.grid(row=4, column=4, rowspan=3, columnspan=6)  
 Button(root\_window, text='LRU', command=show\_LRU).grid(row=10, column=4)  
 FIFO\_frame.grid(row=4, column=0, rowspan=3, columnspan=7)  
 LRU\_frame.grid(row=4, column=4, rowspan=2, columnspan=7)  
 frame2.grid(row=4, column=0, rowspan=6, columnspan=7)  
 ###################################################################################  
 root\_window.mainloop()  
  
  
*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* FIFO(*pages*, *memory\_size*):  
 FIFO\_page = []  
 FIFO\_delete = []  
 memory = []  
 error\_num = 0  
 pages\_length = len(*pages*)  
 *for* page *in pages*:  
 *if* page *not in* memory:  
 error\_num += 1  
 # print(memory)  
 *if* len(memory) < *memory\_size*:  
 memory.append(page)  
 FIFO\_delete.append(0)  
 *else*:  
 memory.append(page)  
 FIFO\_delete.append(memory[0])  
 *del* memory[0]  
 *else*:  
 FIFO\_delete.append(0)  
 FIFO\_page.append(memory.copy())  
 *return* FIFO\_delete, FIFO\_page, error\_num, error\_num/pages\_length  
*def* LRU(*pages*, *memory\_size*):  
 LRU\_page = []  
 LRU\_delete = []  
 memory = []  
 error\_num = 0  
 pages\_length = len(*pages*)  
 *for* page *in pages*:  
 *if* page *not in* memory:  
 error\_num += 1  
 *if* len(memory) < *memory\_size*:  
 memory.append(page)  
 LRU\_delete.append(0)  
 *else*:  
 memory.append(page)  
 LRU\_delete.append(memory[0])  
 *del* memory[0]  
 *else*:  
 k = memory.index(page)  
 temp = memory[k]  
 LRU\_delete.append(temp)  
 *del* memory[k]  
 memory.append(temp)  
 LRU\_page.append(memory.copy())  
 *return* LRU\_delete, LRU\_page, error\_num, error\_num/pages\_length  
  
  
  
  
  
*if* \_\_name\_\_ == '\_\_main\_\_':  
 pages = [1,22,3,4,23,32,33]  
 GUI()  
 # print(FIFO(pages, 4))