北京工业大学

2020 - 2021 学年 第1学期

信息学部 计算机学院

|  |  |  |  |
| --- | --- | --- | --- |
| 课程名称： | 数据结构课程设计 | | |
| 报告性质： | 实验报告 | | |
| 学号： | 18074102 | 姓名： | 左帅 |
| 任课教师： | 王众 | 课程性质： | 学科基础必修课 |
| 学分： | 2 | 学时： | 60 |
| 班级： | 180701 | 成绩： |  |
| 教师评语： |  | | |

|  |  |  |
| --- | --- | --- |
| 需求分析 | 根据题中需求，提供功能划分说明 |  |
| 设计 | 逻辑结构、存储结构设计、算法描述 |  |
| 使用说明 | 界面是否友好 |  |
| 总结 | 是否感悟有收获 |  |
| 摘要 | 考察文字抽象能力 |  |
| 格式 | 是否有目录、页号 |  |

2020年 12 月 17 日

目录

[1需求分析](#_Toc2548)

[1.1功能概述](#_Toc21636)

[1.1.1基础功能](#_Toc8209)

[1.1.2拓展功能](#_Toc3870)

[1.2需要处理的数据](#_Toc4743)

[1.3程序开发运行环境](#_Toc1422)

[1.4用户界面设计](#_Toc10209)

[2数据结构设计](#_Toc13603)

[2.1主要数据结构定义](#_Toc15707)

[2.1.1逻辑结构](#_Toc11947)

[2.1.2存储结构](#_Toc12038)

[2.2整体结构及模块功能描述](#_Toc6785)

[2.3各模块功能描述](#_Toc5658)

[2.3.1电影节点类（movie）](#_Toc7718)

[2.3.2电影链表类（Movielist）](#_Toc13998)

[2.4核心算法流程图](#_Toc14209)

[3详细设计](#_Toc118)

[3.1各模块设计](#_Toc824)

[3.1.1 movie](#_Toc25148)

[3.1.2 Movielist](#_Toc20165)

[3.2各模块核心函数](#_Toc28190)

[1.1、交换值函数](#_Toc7212)

[1.2添加评论函数](#_Toc31971)

[2.1判断链表是否为空](#_Toc11400)

[2.2清空链表](#_Toc12522)

[2.3访问节点](#_Toc4033)

[2.4尾插入](#_Toc32116)

[2.4遍历搜索](#_Toc27431)

[2.5快速排序](#_Toc15705)

[4用户手册及测试结果](#_Toc30160)

[1、排序演示](#_Toc31860)

[2、添加评论](#_Toc24800)

[3、 交叉查询演示](#_Toc28990)

[4、 访问网页](#_Toc21294)

[5、初始化](#_Toc20594)

[5、 修改原数据文件](#_Toc6874)

[5总结提高](#_Toc12982)

[5.1体会与收获](#_Toc3284)

[5.2问题与解决](#_Toc17559)

[5.3评价](#_Toc2406)

[6参考文献](#_Toc31581)

# 1需求分析

## 1.1功能概述

### 1.1.1基础功能

本程序需要实现一个电影检索系统，拥有以下功能：

1、设计实现一个图形界面，能根据不同的分类如豆瓣评分、名著改编、地域、剧情、导演、演员、语言查询到想要的电影，并且各种分类之间要有交叉。

2、设计实现用户可以通过输入电影名，直接查询电影

3、可以按照最新、最热排行查询。

### 1.1.2拓展功能

1、可以让用户添加评语

2、可以点击访问外部链接

## 1.2需要处理的数据

针对三个基础功能，对于功能一需要有电影的评分、地域、剧情、导演、演员、语言这几种数据，其中地域,剧情,语言和演员包含多个数据。电影文件存储于filmlist.txt文件中。数据总计250部电影。

## 1.3程序开发运行环境

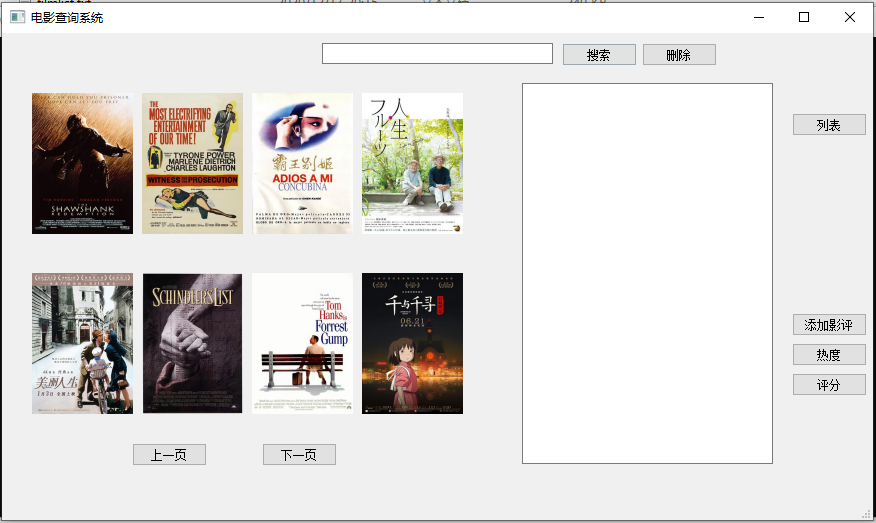
开发环境：

* Visual Studio Code 1.52.1
* Python 3.8.5 64bit

运行环境：

* Windows 10 Professional 64bit 18363

## 1.4用户界面设计



在基础功能1的查找中，用户可以输入电影名，导演名，类别等对电影进行筛选。用户同时也可以点击左边的区域查看电影的信息。所有搜索得到的电影和点击查看的电影热度均会增加。右方的文本框用于信息显示，热度与评分用于对应的排序，添加影评可以与许用户添加影评。

# 2数据结构设计

## 2.1主要数据结构定义

### 2.1.1逻辑结构

对于名称查找使用的是树形结构，同时对于名称和热度排序使用的是x。

### 2.1.2存储结构

电影存储于链表中，选择链表的原因是逻辑简单、操作方便（差删等等），但是在后续中发现链表并不是最优的选择，因为并没有过多的差删操作，更多的访问操作，因此顺序表会更好一点，不过针对链表，在一些算法上做了优化。

为了方便进行快速排序，选择的双向链表。

## 2.2整体结构及模块功能描述

改电影查询系统又电影节点类和电影链表类组成，节点类主要用于存储数据和对数据操作，链表类主要用于实现各种功能。

## 2.3各模块功能描述

### 2.3.1电影节点类（movie）

链表中的节点，负责储存电影的信息。

数据：电影名（name），豆瓣评分（score），名著改编（original），地域（country），语言（language），导演（director），演员（actors），剧情（plot）。

此外为了实现另外两个基础功能，又加入了热度（hot），上架时间（time），两个数据，为了方便后续排序、热度文件写入，又加入了number（按链表顺序从0到n，方便快速排序指针移动，判断指针重合），filenumber（按原电影文件顺序从0-n，用于热度写入）。

为了实现拓展功能，后续又加入了评论（pinglun[]），网页链接（url）

功能 ：

1、交换两个节点数值（用于排序）

2、更新热度

3、添加评论

### 2.3.2电影链表类（Movielist）

链表本体

功能：

1. 按类型查找，各类有交叉
2. 按照时间、热度、默认排序

## 2.4核心算法流程图

开始

读取文件内容

将文件信息储存到AVL树与链表中

是否点击了电影图片

将电影信息，影评，链接显示在信息框

是

否

是否点击了搜索按钮

判断其为电影名或者为其他如导演并进行搜索

是

否

是否点击了热度或评分

根据对应排序方法切换排序好的链表

是

是否点击影评按钮

是

输入搜索的内容

输入影评

为对应的电影添加影评

是否退出

结束

是

否

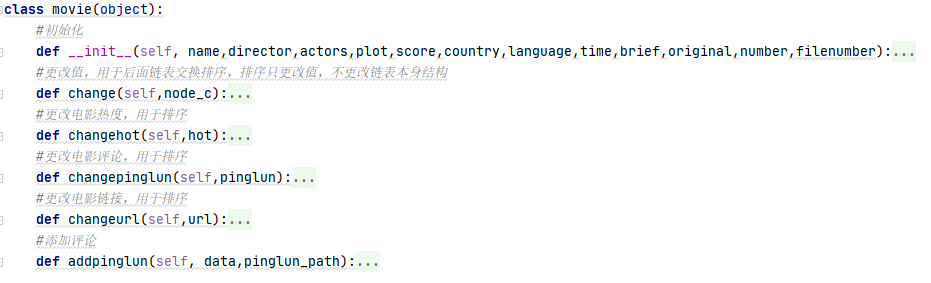
否

+

# 3详细设计

## 3.1各模块设计

### 3.1.1 movie



### 3.1.2 Movielist



## 3.2各模块核心函数

### 1.1、交换值函数

#更改值，用于后面链表交换排序，排序只更改值，不更改链表本身结构

def change(self,node\_c):

self.name = node\_c.name

self.score = node\_c.score

self.country = node\_c.country

self.language = node\_c.language

self.director = node\_c.director

self.actors = node\_c.actors

self.plot = node\_c.plot

self.original = node\_c.original # 改编

self.brief = node\_c.brief # 剧情简介

self.hot = node\_c.hot # 记录点击量

self.time = node\_c.time

self.filenumber = node\_c.filenumber

self.pinglun = node\_c.pinglun

self.url = node\_c.url

#更改电影热度，用于排序

def changehot(self,hot):

self.hot = hot

#更改电影评论，用于排序

def changepinglun(self,pinglun):

self.pinglun = pinglun

#更改电影链接，用于排序

def changeurl(self,url):

self.url = url

该类函数作用是在快速排序和搜索电影功能中。用于快速排序是以为对于链表，交换值的效率要比交换链表的节点效率要高。用于搜索电影是因为每次搜索都会把结果存入一个链表中，下一次搜索会从这个新表中搜索，在创建新表时要用其赋值。

### 1.2添加评论函数

def addpinglun(self, data,pinglun\_path):

if self.pinglun == '':

self.pinglun = list(self.pinglun)

self.pinglun.append(data)

f = open(pinglun\_path, 'r+',encoding='utf-8')

flist = f.readlines()

#newpinglun = str(self.pinglun)

flist[self.filenumber] = data + "/" +flist[self.filenumber]

f = open(pinglun\_path, 'w+',encoding='utf-8')

f.writelines(flist)

f.close()

该函数的作用是添加评论到节点的pinglun[]中，并将pinglun[]写入txt文件中

### 2.1判断链表是否为空

def is\_empty(self):

if self.header == None:

return True

else:

return False

判断链表是否为空，用于插入节点

### 2.2清空链表

def clearlist(self):

self.header = None

self.tail = None

self.length = 0

清空删除表，用于在查询结束后删除查询结果

### 2.3访问节点

def fangwen(self,node,hot\_path):

node.hot = node.hot +1

f = open(hot\_path, 'r+')

flist = f.readlines()

newhot = str(node.hot)

flist[node.filenumber] = newhot+'\n'

f = open(hot\_path, 'w+')

f.writelines(flist)

f.close()

访问、点击电影，热度+1，存入储存热度的文件

### 2.4尾插入

def addend(self,node):

current\_node = self.header

if self.is\_empty():

self.header = node

self.tail = node

else:

while current\_node.next != None:

current\_node=current\_node.next

current\_node.next = node

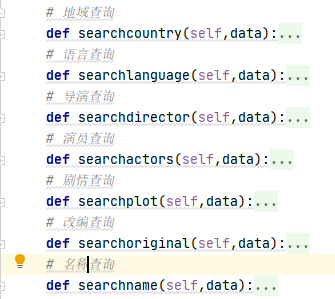
node.before = current\_node

self.tail = node

self.length += 1

在链表结尾插入节点

### 2.4遍历搜索



def searchcountry(self,data):

current\_Node = self.header # 头指针，用于遍历当前链表

empty\_list = MovieList() # 空表，存储这次检索出的信息

#data = input("请输入剧情地域:")

len = 0

while len <= self.length:

if data in current\_Node.country:

# node用于暂存符合检索条件的current\_node的信息，但是其next=None

node = movie(current\_Node.name, current\_Node.director, current\_Node.actors, current\_Node.plot,

current\_Node.score, current\_Node.country, current\_Node.language, current\_Node.time,

current\_Node.brief, current\_Node.original,current\_Node.number,current\_Node.filenumber)

node.changehot(current\_Node.hot)

node.changepinglun(current\_Node.pinglun)

node.changeurl(current\_Node.url)

empty\_list.addend(node) # 将符合检索信息的节点加入空链表

current\_Node = current\_Node.next # 检索当前表下一个节点

else:

current\_Node = current\_Node.next # 检索当前表下一个节点

len += 1

print("查询结束")

self.length = empty\_list.length

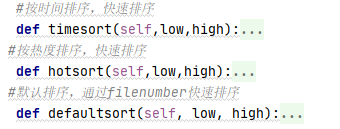
self.header = empty\_list.header

self.tail = empty\_list.tail

遍历搜索包括遍历搜索地域、语言、导演、演员、剧情、名字、是否改编，每个搜索函数的机构类似，只是更改了搜索的数据类型。

此处以遍历搜索地域为例子。其逻辑为遍历链表，当当前节点的值等于搜索的Data值时，将改节点的值复制到新链表中，并在最后将这个新链表代替当前链表进行下一次的搜索。

### 2.5快速排序



def timesort(self,low,high):

key = movie('abc', 'abc', 'abc', 'abc', 'abc', 'abc', 'abc', 'abc', 'abc','no',-1,-1)

if not low or not low.next:

return

if low.number != high.number:

p=low

q=high

key.change(p)

while p.number != q.number:

while p.number != q.number and q.time <= key.time:

q = q.before

p.change(q)

while p.number != q.number and p.time >= key.time:

p = p.next

q.change(p)

q.change(key)

if low.time != p.time:

self.timesort(low,p.before)

if p.time != high.time:

self.timesort(p.next,high)

排序分为时间排序、热度排序、默认排序，采用的都是快速排序的思路（O (nlogn)）。两个指针，p是头、q是尾，在交换节点时，不对链表本身进行改动，只是交换数据（调用节点中的change（）函数），这样稍微弥补了一些链表本身交换位置效率低的缺陷。

热度排序与默认排序与时间排序结构相同，只是排序所根据的值换成了hot和filenumber（文件中的顺序号）

## 3.3详细代码

### 1.1 AVLTree.py

from Film import film

from linkedlist import linked\_list

class AVLTree:

    class film\_node:

        def \_\_init\_\_(self, film):

            self.film\_name = film.name

            self.left = None

            self.right = None

            self.height = 1

            self.film = film

            self.strs=''

    def \_\_init\_\_(self):

        self.root = None

        self.size = 0

        self.need\_refresh\_list=[]

    def get\_size(self):

        return self.size

    def add\_from\_root(self,film):

        self.root=self.add(self.root,film)

    def add(self, node,film):

        if not node:

            self.size += 1#若节点为空，则创建一个节点

            return self.film\_node(film)

        elif node.film\_name > film.name:

            node.left = self.add(node.left,film)

        else:

            node.right = self.add(node.right,film)

        # 更新height,获取平衡因子

        node.height = 1 + max(self.get\_height(node.left), self.get\_height(node.right))

        banlance\_factor = self.get\_balance\_factor(node)

        # 维护平衡性，共四种情况LL,RR,LR,RL

        #LL

        if banlance\_factor > 1 and self.get\_balance\_factor(node.left) >= 0:

            return self.right\_rotate(node)

        #RR

        if banlance\_factor < -1 and self.get\_balance\_factor(node.right) <= 0:

            return self.left\_rotate(node)

        #LR

        if banlance\_factor > 1 and self.get\_balance\_factor(node.left) < 0:

            node.left = self.left\_rotate(node.left)

            return self.right\_rotate(node)

        #RL

        if banlance\_factor < -1 and self.get\_balance\_factor(node.right) > 0:

            node.right = self.right\_rotate(node.right)

            return self.left\_rotate(node)

        return node

    def get\_height(self, node):

        if not node:

            return 0

        return node.height

    def get\_balance\_factor(self, node):

        if not node:

            return 0

        return self.get\_height(node.left) - self.get\_height(node.right)

    def right\_rotate(self, y):

        x = y.left

        T3 = x.right

        # 右旋转

        x.right = y

        y.left = T3

        # 更新height

        y.height = max(self.get\_height(y.left), self.get\_height(y.right)) + 1

        x.height = max(self.get\_height(x.left), self.get\_height(x.right)) + 1

        return x

    def left\_rotate(self, y):

        x = y.right

        T2 = x.left

        x.left = y

        y.right = T2

        # 更新height

        y.height = max(self.get\_height(y.left), self.get\_height(y.right)) + 1

        x.height = max(self.get\_height(x.left), self.get\_height(x.right)) + 1

        return x

    def get\_node(self, node, film\_name):

        if not node:

            return

        if node.film\_name == film\_name:

            return node

        elif node.film\_name > film\_name:

            return self.get\_node(node.left, film\_name)

        else:

            return self.get\_node(node.right, film\_name)

    def get\_film\_from\_node(self,film\_name):

        s = self.get\_node(self.root, film\_name)

        if s is not None:

            return s.film

        else:

            return film('noname','norating','noregion','nointro','notypes','nosuchdirector','noactor','nolanguage','notime','nourl','nope')

    def preorder\_traversal\_save(self):

        film=open("filmlist.txt", "w",encoding='utf-8')

        self.\_preorder\_traversal\_save(self.root,film)

        film.close()

    def \_preorder\_traversal\_save(self,node,film):

        if node is not None:

            #filmname,director,actors,types,ratings,regions,languages,date,intro

            film.write(node.film.name)

            film.write('#')

            film.write(node.film.director)

            film.write('#')

            film.write('/'.join(node.film.actors))

            film.write('#')

            film.write('/'.join(node.film.types))

            film.write('#')

            film.write(node.film.rating)

            film.write('#')

            film.write('/'.join(node.film.region))

            film.write('#')

            film.write('/'.join(node.film.language))

            film.write('#')

            film.write(node.film.time)

            film.write('#')

            film.write(node.film.plot)

            film.write('#')

            film.write(node.film.url)

            film.write('#')

            film.write(str(node.film.click\_times))

            film.write('#')

            film.write('/'.join(node.film.review))

            film.write('\n')

            self.\_preorder\_traversal\_save(node.left,film)

            self.\_preorder\_traversal\_save(node.right,film)

    def get\_film\_from\_node\_llist(self,film\_name):

        s = self.get\_node(self.root, film\_name)

        llist = linked\_list()

        self.need\_refresh\_list=[]

        if s is not None:

            llist.append(s.film)

            s.film.clicked()

            self.need\_refresh\_list=[s]

            return llist

        else:

            return film('noname','norating','noregion','nointro','notypes','nosuchdirector','noactor','nolanguage','notime', 'nourl','nope')

    def get\_film\_from\_node\_print(self,film\_name):

        self.searched\_film = self.get\_film\_from\_node(film\_name)

        if self.searched\_film.name != 'noname':

            print(self.searched\_film.name,self.searched\_film.director,self.searched\_film.types)

            self.searched\_film.clicked()

            self.need\_refresh\_list=[self.searched\_film]

            return '电影名称:'+self.searched\_film.name+ \

            '\n导演:'+self.searched\_film.director+ \

            '\n演员:'+' '.join(self.searched\_film.actors)+ \

            '\n类型:'+' '.join(self.searched\_film.types)+ \

            '\n上映日期: '+self.searched\_film.time+ \

            '\n语言:'+' '.join(self.searched\_film.language)+ \

            '\n区域:'+' '.join(self.searched\_film.region)+ \

            '\n评分:'+self.searched\_film.rating+ \

            '\n剧情梗概:'+self.searched\_film.plot+'\n'

        else:

            print('No Such Film!')

            return 'No Such Film!'

    def contains(self, film\_name):

        return self.get\_node(self.root, film\_name) is not None

    def preorder\_traversal(self,node):

        if node is not None:

            print(node.film\_name,node.film.director,node.film.types)

            self.strs=self.strs+node.film\_name+'  ['+' '.join(node.film.types)+']\n'

            self.preorder\_traversal(node.left)

            self.preorder\_traversal(node.right)

    def root\_preorder\_traversal(self):

        self.strs=''

        self.preorder\_traversal(self.root)

        return self.strs

    def root\_preorder\_traversal\_search(self,types):

        self.flag = 0

        self.strs = ''

        self.llist = linked\_list()

        self.need\_refresh\_list=[]

        self.preorder\_traversal\_search(self.root,types)

        if self.flag == 0:

            print("No Such Film!")

            return 'No Such Film!'

        #return self.strs

        return self.llist

    def preorder\_traversal\_search(self,node,search\_str):

        if node is not None:

            i = 1

            strs=search\_str.split()

            for string in strs:

                if node.film.types.count(string) < 1 and node.film.director!=string and node.film.actors.count(string) < 1:

                    i = 0

            if i == 1:

                self.flag = 1

                #print(node.film\_name,node.film.director,node.film.actors,node.film.types)

                node.film.clicked()

                self.need\_refresh\_list.append(node)

                self.llist.append(node.film)

                #self.strs =self.strs + '电影名称：'+node.film.name+ \

                #'\n导演:'+node.film.director+ \

                #'\n演员:'+' '.join(node.film.actors)+ \

                #'\n类型:'+' '.join(node.film.types)+ \

                #'\n上映日期: '+node.film.time+ \

                #'\n语言:'+' '.join(node.film.language)+ \

                #'\n区域:'+' '.join(node.film.region)+ \

                #'\n评分:'+node.film.rating+ \

                #'\n剧情梗概:'+node.film.plot+'\n\n'

            self.preorder\_traversal\_search(node.left,search\_str)

            self.preorder\_traversal\_search(node.right,search\_str)

    def remove(self, film\_name):

        node = self.get\_node(self.root, film\_name)

        if node:

            self.root = self.\_remove(self.root, film\_name)

        # 删除以node为根的BST中键值为key的节点，递归算法

        # 返回删除节点后的新的BSTMap的根

    def minimum(self, node):

        if not node.left:

            return node

        return self.minimum(node.left)

    def \_remove(self, node, film\_name):

        if not node:

            return

        # 递归条件

        if node.film\_name > film\_name:

            node.left = self.\_remove(node.left, film\_name)

            ret\_node = node

        elif node.film\_name < film\_name:

            node.right = self.\_remove(node.right, film\_name)

            ret\_node = node

        else:  # node.film\_name == film\_name

            if not node.left:

                right\_node = node.right

                node.right = None

                self.size -= 1

                ret\_node = right\_node

            elif not node.right:

                left\_node = node.left

                node.left = None

                self.size -= 1

                ret\_node = left\_node

            else:

                successor = self.minimum(node.right)

                successor.right = self.\_remove(node.right, successor.film\_name)

                successor.left = node.left

                node.left = node.right = None

                ret\_node = successor

        if not ret\_node:

            return

            # 需要更新height

        ret\_node.height = 1 + max(

            self.get\_height(ret\_node.left),

            self.get\_height(ret\_node.right),

        )

        banlance\_factor = self.get\_balance\_factor(ret\_node)

        #LL

        if banlance\_factor > 1 and self.get\_balance\_factor(ret\_node.left) >= 0:

            return self.right\_rotate(ret\_node)

        #RR

        if banlance\_factor < -1 and self.get\_balance\_factor(ret\_node.right) <= 0:

            return self.left\_rotate(ret\_node)

        # LR

        if banlance\_factor > 1 and self.get\_balance\_factor(ret\_node.left) < 0:

            ret\_node.left = self.left\_rotate(ret\_node.left)

            return self.right\_rotate(ret\_node)

        # RL

        if banlance\_factor < -1 and self.get\_balance\_factor(ret\_node.right) > 0:

            ret\_node.right = self.right\_rotate(ret\_node.right)

            return self.left\_rotate(ret\_node)

        return ret\_node

### 1.2 Film.py

class film():

    def \_\_init\_\_(self,name,director,actors,types,rating,region,language,time,plot,url,click\_times=0,review='noreview'):

        #filmname,director,actors,types,ratings,regions,languages,date,intro

    #def \_\_init\_\_(self,name,director,types):

        self.name = name

        self.rating = rating

        #self.adapt\_from = adapt\_from

        self.region = region.split('/')

        self.types = types.split('/')

        self.plot = plot

        self.director = director

        self.actors = actors.split('/')

        self.language = language.split('/')

        self.time = time

        #self.types = types

        self.click\_times = int(click\_times)

        self.image='./img/'+self.name+'.webp'

        self.url=url

        self.review=review.split('/')

    def clicked(self):

        self.click\_times = self.click\_times + 1

### 1.3 GUI.py

import sys

from PyQt5.QtWidgets import QApplication, QMainWindow, QMessageBox

from PyQt5 import QtCore

from Ui\_GUI import \*

from Ui\_review import \*

from search import Search

import webbrowser

class MyQLabel(QtWidgets.QLabel):

    button\_clicked\_signal = QtCore.pyqtSignal()

    def \_\_init\_\_(self, parent=None):

        super(MyQLabel, self).\_\_init\_\_(parent)

        self.node=None

    def mouseReleaseEvent(self, QMouseEvent):

        self.button\_clicked\_signal.emit()

    def connect\_customized\_slot(self, func):

        self.button\_clicked\_signal.connect(func)

class Review(QtWidgets.QMainWindow, Ui\_review):

    def \_\_init\_\_(self, parent=None):

        super(Review, self).\_\_init\_\_(parent)

        self.setupUi(self)

        self.review=''

    def ConfirmClicked(self):

        self.review=self.Enter.toPlainText()

        self.my\_Signal.emit(self.review)

        self.close()

    def CancelClicked(self):

        self.close()

    my\_Signal = QtCore.pyqtSignal(str)

class MyWindow(QMainWindow, Ui\_FSS):

    def \_\_init\_\_(self, parent=None):

        super(MyWindow, self).\_\_init\_\_(parent)

        self.s=Search()

        self.setFocusPolicy(QtCore.Qt.StrongFocus)

        self.rev=Review()

        self.setFocus()

#        self.SearchButton.clicked.connect(self.SearchButtonClicked)

        self.setupUi(self)

        self.lastimg=None

        self.allhead=None

        self.init\_pages()

        self.sym=1

        self.current\_url=''

        self.currentfilm=None

    my\_Signal = QtCore.pyqtSignal(str)

    def eventFilter(self, obj, event):

        #if obj == self.SearchText:

        if event.type() == QtCore.QEvent.KeyPress and event.key() == QtCore.Qt.Key\_Enter:

            self.SearchButtonClicked()

            return True

        return QtCore.QObject.eventFilter(self, obj, event)

    def SearchButtonClicked(self):

        strs=self.s.search(self.SearchText.text())

        if strs=='No Such Film!':

            self.imformation.setText(strs)

            QMessageBox.question(self, '错误',"未找到符合条件的电影",QMessageBox.Ok)

            return

        #self.imformation.setText(strs)

        self.allhead=strs.head

        self.lastimg=strs.head

        pointer=self.lastimg

        self.sym=4

        self.img1.node=pointer

        if pointer is not None:

            self.img1.setPixmap(QtGui.QPixmap(pointer.film.image))

            pointer=pointer.next

        else:

            self.img1.setPixmap(QtGui.QPixmap('./img/timg.png'))

        self.img2.node=pointer

        if pointer is not None:

            self.img2.setPixmap(QtGui.QPixmap(pointer.film.image))

            pointer=pointer.next

        else:

            self.img2.setPixmap(QtGui.QPixmap('./img/timg.png'))

        self.img3.node=pointer

        if pointer is not None:

            self.img3.setPixmap(QtGui.QPixmap(pointer.film.image))

            pointer=pointer.next

        else:

            self.img3.setPixmap(QtGui.QPixmap('./img/timg.png'))

        self.img4.node=pointer

        if pointer is not None:

            self.img4.setPixmap(QtGui.QPixmap(pointer.film.image))

            pointer=pointer.next

        else:

            self.img4.setPixmap(QtGui.QPixmap('./img/timg.png'))

        self.img5.node=pointer

        if pointer is not None:

            self.img5.setPixmap(QtGui.QPixmap(pointer.film.image))

            pointer=pointer.next

        else:

            self.img5.setPixmap(QtGui.QPixmap('./img/timg.png'))

        self.img6.node=pointer

        if pointer is not None:

            self.img6.setPixmap(QtGui.QPixmap(pointer.film.image))

            pointer=pointer.next

        else:

            self.img6.setPixmap(QtGui.QPixmap('./img/timg.png'))

        self.img7.node=pointer

        if pointer is not None:

            self.img7.setPixmap(QtGui.QPixmap(pointer.film.image))

            pointer=pointer.next

        else:

            self.img7.setPixmap(QtGui.QPixmap('./img/timg.png'))

        if pointer is not None:

            self.img8.setPixmap(QtGui.QPixmap(pointer.film.image))

        else:

            self.img8.setPixmap(QtGui.QPixmap('./img/timg.png'))

        self.img8.node=pointer

        if pointer is not None:

            self.lastimg=pointer.next

    def DeleteButtonClicked(self):

        if self.s.avl.contains(self.SearchText.text()):

            reply=QMessageBox.question(self, '提示',"您确定要删除吗?", QMessageBox.Yes | QMessageBox.No, QMessageBox.No)

            if reply==QMessageBox.Yes:

                self.s.delete(self.SearchText.text())

                QMessageBox.question(self, '成功',"已删除",QMessageBox.Ok)

        else:

            self.imformation.setText('No Such Film!')

            QMessageBox.question(self, '错误',"未找到符合条件的电影",QMessageBox.Ok)

            return

    def init\_pages(self):

        pointer=self.s.rating\_sort.head.filmlist.head

        self.allhead=pointer

        self.img1.node=pointer

        if pointer is not None:

            self.img1.setPixmap(QtGui.QPixmap(pointer.film.image))

            pointer=self.s.rating\_sort.gnext(pointer)

        else:

            self.img1.setPixmap(QtGui.QPixmap('./img/timg.png'))

        self.img2.node=pointer

        if pointer is not None:

            self.img2.setPixmap(QtGui.QPixmap(pointer.film.image))

            pointer=self.s.rating\_sort.gnext(pointer)

        else:

            self.img2.setPixmap(QtGui.QPixmap('./img/timg.png'))

        self.img3.node=pointer

        if pointer is not None:

            self.img3.setPixmap(QtGui.QPixmap(pointer.film.image))

            pointer=self.s.rating\_sort.gnext(pointer)

        else:

            self.img3.setPixmap(QtGui.QPixmap('./img/timg.png'))

        self.img4.node=pointer

        if pointer is not None:

            self.img4.setPixmap(QtGui.QPixmap(pointer.film.image))

            pointer=self.s.rating\_sort.gnext(pointer)

        else:

            self.img4.setPixmap(QtGui.QPixmap('./img/timg.png'))

        self.img5.node=pointer

        if pointer is not None:

            self.img5.setPixmap(QtGui.QPixmap(pointer.film.image))

            pointer=self.s.rating\_sort.gnext(pointer)

        else:

            self.img5.setPixmap(QtGui.QPixmap('./img/timg.png'))

        self.img6.node=pointer

        if pointer is not None:

            self.img6.setPixmap(QtGui.QPixmap(pointer.film.image))

            pointer=self.s.rating\_sort.gnext(pointer)

        else:

            self.img6.setPixmap(QtGui.QPixmap('./img/timg.png'))

        self.img7.node=pointer

        if pointer is not None:

            self.img7.setPixmap(QtGui.QPixmap(pointer.film.image))

            pointer=self.s.rating\_sort.gnext(pointer)

        else:

            self.img7.setPixmap(QtGui.QPixmap('./img/timg.png'))

        if pointer is not None:

            self.img8.setPixmap(QtGui.QPixmap(pointer.film.image))

        else:

            self.img8.setPixmap(QtGui.QPixmap('./img/timg.png'))

        self.img8.node=pointer

        if pointer is not None:

            self.lastimg=self.s.rating\_sort.gnext(pointer)

    def NextPageClicked(self):

        pointer=self.lastimg

        self.img1.node=pointer

        if self.sym == 1:

            if pointer is not None:

                self.img1.setPixmap(QtGui.QPixmap(pointer.film.image))

                pointer=self.s.rating\_sort.gnext(pointer)

            else:

                self.img1.setPixmap(QtGui.QPixmap('./img/timg.png'))

            self.img2.node=pointer

            if pointer is not None:

                self.img2.setPixmap(QtGui.QPixmap(pointer.film.image))

                pointer=self.s.rating\_sort.gnext(pointer)

            else:

                self.img2.setPixmap(QtGui.QPixmap('./img/timg.png'))

            self.img3.node=pointer

            if pointer is not None:

                self.img3.setPixmap(QtGui.QPixmap(pointer.film.image))

                pointer=self.s.rating\_sort.gnext(pointer)

            else:

                self.img3.setPixmap(QtGui.QPixmap('./img/timg.png'))

            self.img4.node=pointer

            if pointer is not None:

                self.img4.setPixmap(QtGui.QPixmap(pointer.film.image))

                pointer=self.s.rating\_sort.gnext(pointer)

            else:

                self.img4.setPixmap(QtGui.QPixmap('./img/timg.png'))

            self.img5.node=pointer

            if pointer is not None:

                self.img5.setPixmap(QtGui.QPixmap(pointer.film.image))

                pointer=self.s.rating\_sort.gnext(pointer)

            else:

                self.img5.setPixmap(QtGui.QPixmap('./img/timg.png'))

            self.img6.node=pointer

            if pointer is not None:

                self.img6.setPixmap(QtGui.QPixmap(pointer.film.image))

                pointer=self.s.rating\_sort.gnext(pointer)

            else:

                self.img6.setPixmap(QtGui.QPixmap('./img/timg.png'))

            self.img7.node=pointer

            if pointer is not None:

                self.img7.setPixmap(QtGui.QPixmap(pointer.film.image))

                pointer=self.s.rating\_sort.gnext(pointer)

            else:

                self.img7.setPixmap(QtGui.QPixmap('./img/timg.png'))

            if pointer is not None:

                self.img8.setPixmap(QtGui.QPixmap(pointer.film.image))

            else:

                self.img8.setPixmap(QtGui.QPixmap('./img/timg.png'))

            self.img8.node=pointer

            if pointer is not None:

                self.lastimg=self.s.rating\_sort.gnext(pointer)

        elif self.sym==2:

            if pointer is not None:

                self.img1.setPixmap(QtGui.QPixmap(pointer.film.image))

                pointer=self.s.clicks\_sort.gnext(pointer)

            else:

                self.img1.setPixmap(QtGui.QPixmap('./img/timg.png'))

            self.img2.node=pointer

            if pointer is not None:

                self.img2.setPixmap(QtGui.QPixmap(pointer.film.image))

                pointer=self.s.clicks\_sort.gnext(pointer)

            else:

                self.img2.setPixmap(QtGui.QPixmap('./img/timg.png'))

            self.img3.node=pointer

            if pointer is not None:

                self.img3.setPixmap(QtGui.QPixmap(pointer.film.image))

                pointer=self.s.clicks\_sort.gnext(pointer)

            else:

                self.img3.setPixmap(QtGui.QPixmap('./img/timg.png'))

            self.img4.node=pointer

            if pointer is not None:

                self.img4.setPixmap(QtGui.QPixmap(pointer.film.image))

                pointer=self.s.clicks\_sort.gnext(pointer)

            else:

                self.img4.setPixmap(QtGui.QPixmap('./img/timg.png'))

            self.img5.node=pointer

            if pointer is not None:

                self.img5.setPixmap(QtGui.QPixmap(pointer.film.image))

                pointer=self.s.clicks\_sort.gnext(pointer)

            else:

                self.img5.setPixmap(QtGui.QPixmap('./img/timg.png'))

            self.img6.node=pointer

            if pointer is not None:

                self.img6.setPixmap(QtGui.QPixmap(pointer.film.image))

                pointer=self.s.clicks\_sort.gnext(pointer)

            else:

                self.img6.setPixmap(QtGui.QPixmap('./img/timg.png'))

            self.img7.node=pointer

            if pointer is not None:

                self.img7.setPixmap(QtGui.QPixmap(pointer.film.image))

                pointer=self.s.clicks\_sort.gnext(pointer)

            else:

                self.img7.setPixmap(QtGui.QPixmap('./img/timg.png'))

            if pointer is not None:

                self.img8.setPixmap(QtGui.QPixmap(pointer.film.image))

            else:

                self.img8.setPixmap(QtGui.QPixmap('./img/timg.png'))

            self.img8.node=pointer

            if pointer is not None:

                self.lastimg=self.s.clicks\_sort.gnext(pointer)

        elif self.sym == 4:

            if pointer is not None:

                self.img1.setPixmap(QtGui.QPixmap(pointer.film.image))

                pointer=pointer.next

            else:

                self.img1.setPixmap(QtGui.QPixmap('./img/timg.png'))

            self.img2.node=pointer

            if pointer is not None:

                self.img2.setPixmap(QtGui.QPixmap(pointer.film.image))

                pointer=pointer.next

            else:

                self.img2.setPixmap(QtGui.QPixmap('./img/timg.png'))

            self.img3.node=pointer

            if pointer is not None:

                self.img3.setPixmap(QtGui.QPixmap(pointer.film.image))

                pointer=pointer.next

            else:

                self.img3.setPixmap(QtGui.QPixmap('./img/timg.png'))

            self.img4.node=pointer

            if pointer is not None:

                self.img4.setPixmap(QtGui.QPixmap(pointer.film.image))

                pointer=pointer.next

            else:

                self.img4.setPixmap(QtGui.QPixmap('./img/timg.png'))

            self.img5.node=pointer

            if pointer is not None:

                self.img5.setPixmap(QtGui.QPixmap(pointer.film.image))

                pointer=pointer.next

            else:

                self.img5.setPixmap(QtGui.QPixmap('./img/timg.png'))

            self.img6.node=pointer

            if pointer is not None:

                self.img6.setPixmap(QtGui.QPixmap(pointer.film.image))

                pointer=pointer.next

            else:

                self.img6.setPixmap(QtGui.QPixmap('./img/timg.png'))

            self.img7.node=pointer

            if pointer is not None:

                self.img7.setPixmap(QtGui.QPixmap(pointer.film.image))

                pointer=pointer.next

            else:

                self.img7.setPixmap(QtGui.QPixmap('./img/timg.png'))

            if pointer is not None:

                self.img8.setPixmap(QtGui.QPixmap(pointer.film.image))

            else:

                self.img8.setPixmap(QtGui.QPixmap('./img/timg.png'))

            self.img8.node=pointer

            if pointer is not None:

                self.lastimg=pointer.next

    def Show\_film(self):

        send = self.sender()

        self.currentfilm=send.node.film

        if send.node is not None:

            send.node.film.clicked()

            self.s.refresh\_list\_external(self.s.clicks\_sort,send.node.film)

            \_translate = QtCore.QCoreApplication.translate

            self.current\_url=send.node.film.url

            if send.node.film.review[0]=='noreview':

                self.imformation.setHtml('电影名称:'+send.node.film.name+'<br />'+\

                    '导演:'+send.node.film.director+'<br />'+\

                    '演员:'+' '.join(send.node.film.actors)+'<br />'+\

                    '类型:'+' '.join(send.node.film.types)+'<br />'+\

                    '上映日期: '+send.node.film.time+'<br />'+\

                    '语言:'+' '.join(send.node.film.language)+'<br />'+\

                    '区域:'+' '.join(send.node.film.region)+'<br />'+\

                    '评分:'+send.node.film.rating+'<br />'+\

                    '剧情梗概:'+send.node.film.plot+'<br />'+\

                    '<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><a href=\"'+\

                    send.node.film.url+\

                    '\"><span style=\" text-decoration: underline; color:#0000ff;\">'+\

                    send.node.film.url+'</span></a></p>')

            else:

                self.imformation.setHtml('电影名称:'+send.node.film.name+'<br />'+\

                    '导演:'+send.node.film.director+'<br />'+\

                    '演员:'+' '.join(send.node.film.actors)+'<br />'+\

                    '类型:'+' '.join(send.node.film.types)+'<br />'+\

                    '上映日期: '+send.node.film.time+'<br />'+\

                    '语言:'+' '.join(send.node.film.language)+'<br />'+\

                    '区域:'+' '.join(send.node.film.region)+'<br />'+\

                    '评分:'+send.node.film.rating+'<br />'+\

                    '剧情梗概:'+send.node.film.plot+'<br />'+\

                    '影评:<br />'+'<br />'.join(send.node.film.review)+'<br />'+\

                    '<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><a href=\"'+\

                    send.node.film.url+\

                    '\"><span style=\" text-decoration: underline; color:#0000ff;\">'+\

                    send.node.film.url+'</span></a></p>')

        else:

            #self.imformation.setText('error')

            pass

    def FrontPageClicked(self):

        pointer=self.allhead

        if pointer!=self.lastimg:

            tmp1=pointer

            tmp=pointer

            while pointer is not None:

                tmp1=tmp

                tmp=pointer

                if self.sym == 1:

                    if pointer is not None:

                        pointer=self.s.rating\_sort.gnext(pointer)

                    if pointer is not None:

                        pointer=self.s.rating\_sort.gnext(pointer)

                    if pointer is not None:

                        pointer=self.s.rating\_sort.gnext(pointer)

                    if pointer is not None:

                        pointer=self.s.rating\_sort.gnext(pointer)

                    if pointer is not None:

                        pointer=self.s.rating\_sort.gnext(pointer)

                    if pointer is not None:

                        pointer=self.s.rating\_sort.gnext(pointer)

                    if pointer is not None:

                        pointer=self.s.rating\_sort.gnext(pointer)

                    if pointer is not None:

                        pointer=self.s.rating\_sort.gnext(pointer)

                    if pointer == self.lastimg:

                        pointer=tmp1

                        if pointer == None:

                            pointer=tmp

                        if self.img1.node == self.lastimg:

                            pointer=tmp

                        self.img1.node=pointer

                        if pointer is not None:

                            self.img1.setPixmap(QtGui.QPixmap(pointer.film.image))

                            pointer=self.s.rating\_sort.gnext(pointer)

                        else:

                            self.img1.setPixmap(QtGui.QPixmap('./img/timg.png'))

                        self.img2.node=pointer

                        if pointer is not None:

                            self.img2.setPixmap(QtGui.QPixmap(pointer.film.image))

                            pointer=self.s.rating\_sort.gnext(pointer)

                        else:

                            self.img2.setPixmap(QtGui.QPixmap('./img/timg.png'))

                        self.img3.node=pointer

                        if pointer is not None:

                            self.img3.setPixmap(QtGui.QPixmap(pointer.film.image))

                            pointer=self.s.rating\_sort.gnext(pointer)

                        else:

                            self.img3.setPixmap(QtGui.QPixmap('./img/timg.png'))

                        self.img4.node=pointer

                        if pointer is not None:

                            self.img4.setPixmap(QtGui.QPixmap(pointer.film.image))

                            pointer=self.s.rating\_sort.gnext(pointer)

                        else:

                            self.img4.setPixmap(QtGui.QPixmap('./img/timg.png'))

                        self.img5.node=pointer

                        if pointer is not None:

                            self.img5.setPixmap(QtGui.QPixmap(pointer.film.image))

                            pointer=self.s.rating\_sort.gnext(pointer)

                        else:

                            self.img5.setPixmap(QtGui.QPixmap('./img/timg.png'))

                        self.img6.node=pointer

                        if pointer is not None:

                            self.img6.setPixmap(QtGui.QPixmap(pointer.film.image))

                            pointer=self.s.rating\_sort.gnext(pointer)

                        else:

                            self.img6.setPixmap(QtGui.QPixmap('./img/timg.png'))

                        self.img7.node=pointer

                        if pointer is not None:

                            self.img7.setPixmap(QtGui.QPixmap(pointer.film.image))

                            pointer=self.s.rating\_sort.gnext(pointer)

                        else:

                            self.img7.setPixmap(QtGui.QPixmap('./img/timg.png'))

                        if pointer is not None:

                            self.img8.setPixmap(QtGui.QPixmap(pointer.film.image))

                        else:

                            self.img8.setPixmap(QtGui.QPixmap('./img/timg.png'))

                        self.img8.node=pointer

                        if pointer is not None:

                            self.lastimg=self.s.rating\_sort.gnext(pointer)

                        break

                elif self.sym == 2:

                    if pointer is not None:

                        pointer=self.s.clicks\_sort.gnext(pointer)

                    if pointer is not None:

                        pointer=self.s.clicks\_sort.gnext(pointer)

                    if pointer is not None:

                        pointer=self.s.clicks\_sort.gnext(pointer)

                    if pointer is not None:

                        pointer=self.s.clicks\_sort.gnext(pointer)

                    if pointer is not None:

                        pointer=self.s.clicks\_sort.gnext(pointer)

                    if pointer is not None:

                        pointer=self.s.clicks\_sort.gnext(pointer)

                    if pointer is not None:

                        pointer=self.s.clicks\_sort.gnext(pointer)

                    if pointer is not None:

                        pointer=self.s.clicks\_sort.gnext(pointer)

                    if pointer == self.lastimg:

                        pointer=tmp1

                        if pointer == None:

                            pointer=tmp

                        if self.img1.node == self.lastimg:

                            pointer=tmp

                        self.img1.node=pointer

                        if pointer is not None:

                            self.img1.setPixmap(QtGui.QPixmap(pointer.film.image))

                            pointer=self.s.clicks\_sort.gnext(pointer)

                        else:

                            self.img1.setPixmap(QtGui.QPixmap('./img/timg.png'))

                        self.img2.node=pointer

                        if pointer is not None:

                            self.img2.setPixmap(QtGui.QPixmap(pointer.film.image))

                            pointer=self.s.clicks\_sort.gnext(pointer)

                        else:

                            self.img2.setPixmap(QtGui.QPixmap('./img/timg.png'))

                        self.img3.node=pointer

                        if pointer is not None:

                            self.img3.setPixmap(QtGui.QPixmap(pointer.film.image))

                            pointer=self.s.clicks\_sort.gnext(pointer)

                        else:

                            self.img3.setPixmap(QtGui.QPixmap('./img/timg.png'))

                        self.img4.node=pointer

                        if pointer is not None:

                            self.img4.setPixmap(QtGui.QPixmap(pointer.film.image))

                            pointer=self.s.clicks\_sort.gnext(pointer)

                        else:

                            self.img4.setPixmap(QtGui.QPixmap('./img/timg.png'))

                        self.img5.node=pointer

                        if pointer is not None:

                            self.img5.setPixmap(QtGui.QPixmap(pointer.film.image))

                            pointer=self.s.clicks\_sort.gnext(pointer)

                        else:

                            self.img5.setPixmap(QtGui.QPixmap('./img/timg.png'))

                        self.img6.node=pointer

                        if pointer is not None:

                            self.img6.setPixmap(QtGui.QPixmap(pointer.film.image))

                            pointer=self.s.clicks\_sort.gnext(pointer)

                        else:

                            self.img6.setPixmap(QtGui.QPixmap('./img/timg.png'))

                        self.img7.node=pointer

                        if pointer is not None:

                            self.img7.setPixmap(QtGui.QPixmap(pointer.film.image))

                            pointer=self.s.clicks\_sort.gnext(pointer)

                        else:

                            self.img7.setPixmap(QtGui.QPixmap('./img/timg.png'))

                        if pointer is not None:

                            self.img8.setPixmap(QtGui.QPixmap(pointer.film.image))

                        else:

                            self.img8.setPixmap(QtGui.QPixmap('./img/timg.png'))

                        self.img8.node=pointer

                        if pointer is not None:

                            self.lastimg=self.s.clicks\_sort.gnext(pointer)

                        break

                elif self.sym == 4:

                    if pointer is not None:

                        pointer=pointer.next

                    if pointer is not None:

                        pointer=pointer.next

                    if pointer is not None:

                        pointer=pointer.next

                    if pointer is not None:

                        pointer=pointer.next

                    if pointer is not None:

                        pointer=pointer.next

                    if pointer is not None:

                        pointer=pointer.next

                    if pointer is not None:

                        pointer=pointer.next

                    if pointer is not None:

                        pointer=pointer.next

                    if pointer == self.lastimg:

                        pointer=tmp1

                        if pointer == None:

                            pointer=tmp

                        if self.img1.node == self.lastimg:

                            pointer=tmp

                        self.img1.node=pointer

                        if pointer is not None:

                            self.img1.setPixmap(QtGui.QPixmap(pointer.film.image))

                            pointer=pointer.next

                        else:

                            self.img1.setPixmap(QtGui.QPixmap('./img/timg.png'))

                        self.img2.node=pointer

                        if pointer is not None:

                            self.img2.setPixmap(QtGui.QPixmap(pointer.film.image))

                            pointer=pointer.next

                        else:

                            self.img2.setPixmap(QtGui.QPixmap('./img/timg.png'))

                        self.img3.node=pointer

                        if pointer is not None:

                            self.img3.setPixmap(QtGui.QPixmap(pointer.film.image))

                            pointer=pointer.next

                        else:

                            self.img3.setPixmap(QtGui.QPixmap('./img/timg.png'))

                        self.img4.node=pointer

                        if pointer is not None:

                            self.img4.setPixmap(QtGui.QPixmap(pointer.film.image))

                            pointer=pointer.next

                        else:

                            self.img4.setPixmap(QtGui.QPixmap('./img/timg.png'))

                        self.img5.node=pointer

                        if pointer is not None:

                            self.img5.setPixmap(QtGui.QPixmap(pointer.film.image))

                            pointer=pointer.next

                        else:

                            self.img5.setPixmap(QtGui.QPixmap('./img/timg.png'))

                        self.img6.node=pointer

                        if pointer is not None:

                            self.img6.setPixmap(QtGui.QPixmap(pointer.film.image))

                            pointer=pointer.next

                        else:

                            self.img6.setPixmap(QtGui.QPixmap('./img/timg.png'))

                        self.img7.node=pointer

                        if pointer is not None:

                            self.img7.setPixmap(QtGui.QPixmap(pointer.film.image))

                            pointer=pointer.next

                        else:

                            self.img7.setPixmap(QtGui.QPixmap('./img/timg.png'))

                        if pointer is not None:

                            self.img8.setPixmap(QtGui.QPixmap(pointer.film.image))

                        else:

                            self.img8.setPixmap(QtGui.QPixmap('./img/timg.png'))

                        self.img8.node=pointer

                        if pointer is not None:

                            self.lastimg=pointer.next

                        break

    def DirectorButtonClicked(self):

        strs=self.s.search('directors')

        self.imformation.setText(strs)

    def ActorButtonClicked(self):

        strs=self.s.search('actors')

        self.imformation.setText(strs)

    def ListButtonClicked(self):

        strs=self.s.search('list')

        self.imformation.setText(strs)

    def TypesButtonClicked(self):

        strs=self.s.search('types')

        self.imformation.setText(strs)

    def RatingButtonClicked(self):

        pointer=self.s.rating\_sort.head.filmlist.head

        self.allhead=pointer

        self.img1.node=pointer

        self.sym=1

        if pointer is not None:

            self.img1.setPixmap(QtGui.QPixmap(pointer.film.image))

            pointer=self.s.rating\_sort.gnext(pointer)

        else:

            self.img1.setPixmap(QtGui.QPixmap('./img/timg.png'))

        self.img2.node=pointer

        if pointer is not None:

            self.img2.setPixmap(QtGui.QPixmap(pointer.film.image))

            pointer=self.s.rating\_sort.gnext(pointer)

        else:

            self.img2.setPixmap(QtGui.QPixmap('./img/timg.png'))

        self.img3.node=pointer

        if pointer is not None:

            self.img3.setPixmap(QtGui.QPixmap(pointer.film.image))

            pointer=self.s.rating\_sort.gnext(pointer)

        else:

            self.img3.setPixmap(QtGui.QPixmap('./img/timg.png'))

        self.img4.node=pointer

        if pointer is not None:

            self.img4.setPixmap(QtGui.QPixmap(pointer.film.image))

            pointer=self.s.rating\_sort.gnext(pointer)

        else:

            self.img4.setPixmap(QtGui.QPixmap('./img/timg.png'))

        self.img5.node=pointer

        if pointer is not None:

            self.img5.setPixmap(QtGui.QPixmap(pointer.film.image))

            pointer=self.s.rating\_sort.gnext(pointer)

        else:

            self.img5.setPixmap(QtGui.QPixmap('./img/timg.png'))

        self.img6.node=pointer

        if pointer is not None:

            self.img6.setPixmap(QtGui.QPixmap(pointer.film.image))

            pointer=self.s.rating\_sort.gnext(pointer)

        else:

            self.img6.setPixmap(QtGui.QPixmap('./img/timg.png'))

        self.img7.node=pointer

        if pointer is not None:

            self.img7.setPixmap(QtGui.QPixmap(pointer.film.image))

            pointer=self.s.rating\_sort.gnext(pointer)

        else:

            self.img7.setPixmap(QtGui.QPixmap('./img/timg.png'))

        if pointer is not None:

            self.img8.setPixmap(QtGui.QPixmap(pointer.film.image))

        else:

            self.img8.setPixmap(QtGui.QPixmap('./img/timg.png'))

        self.img8.node=pointer

        if pointer is not None:

            self.lastimg=self.s.rating\_sort.gnext(pointer)

    def HeatButtonClicked(self):

        pointer=self.s.clicks\_sort.head.filmlist.head

        self.allhead=pointer

        self.img1.node=pointer

        self.sym=2

        if pointer is not None:

            self.img1.setPixmap(QtGui.QPixmap(pointer.film.image))

            pointer=self.s.clicks\_sort.gnext(pointer)

        else:

            self.img1.setPixmap(QtGui.QPixmap('./img/timg.png'))

        self.img2.node=pointer

        if pointer is not None:

            self.img2.setPixmap(QtGui.QPixmap(pointer.film.image))

            pointer=self.s.clicks\_sort.gnext(pointer)

        else:

            self.img2.setPixmap(QtGui.QPixmap('./img/timg.png'))

        self.img3.node=pointer

        if pointer is not None:

            self.img3.setPixmap(QtGui.QPixmap(pointer.film.image))

            pointer=self.s.clicks\_sort.gnext(pointer)

        else:

            self.img3.setPixmap(QtGui.QPixmap('./img/timg.png'))

        self.img4.node=pointer

        if pointer is not None:

            self.img4.setPixmap(QtGui.QPixmap(pointer.film.image))

            pointer=self.s.clicks\_sort.gnext(pointer)

        else:

            self.img4.setPixmap(QtGui.QPixmap('./img/timg.png'))

        self.img5.node=pointer

        if pointer is not None:

            self.img5.setPixmap(QtGui.QPixmap(pointer.film.image))

            pointer=self.s.clicks\_sort.gnext(pointer)

        else:

            self.img5.setPixmap(QtGui.QPixmap('./img/timg.png'))

        self.img6.node=pointer

        if pointer is not None:

            self.img6.setPixmap(QtGui.QPixmap(pointer.film.image))

            pointer=self.s.clicks\_sort.gnext(pointer)

        else:

            self.img6.setPixmap(QtGui.QPixmap('./img/timg.png'))

        self.img7.node=pointer

        if pointer is not None:

            self.img7.setPixmap(QtGui.QPixmap(pointer.film.image))

            pointer=self.s.clicks\_sort.gnext(pointer)

        else:

            self.img7.setPixmap(QtGui.QPixmap('./img/timg.png'))

        if pointer is not None:

            self.img8.setPixmap(QtGui.QPixmap(pointer.film.image))

        else:

            self.img8.setPixmap(QtGui.QPixmap('./img/timg.png'))

        self.img8.node=pointer

        if pointer is not None:

            self.lastimg=self.s.clicks\_sort.gnext(pointer)

    def openlinks(self):

        webbrowser.open(self.current\_url)

    def save\_file(self):

        self.s.avl.preorder\_traversal\_save()

    def Show\_review(self):

        self.rev=Review()

        self.rev.my\_Signal.connect(self.AddReview)

        self.rev.show()

    def AddReview(self):

        if self.currentfilm != None:

            if self.currentfilm.review[0]=='noreview':

                self.currentfilm.review=[]

                self.currentfilm.review.append(self.rev.Enter.toPlainText())

            else:

                self.currentfilm.review.append(self.rev.Enter.toPlainText())

    def closeEvent(self, QCloseEvent):

        choice = QMessageBox.question(self, '提示', '您确定要关闭吗？', QMessageBox.Yes | QMessageBox.No)

        if choice == QMessageBox.Yes:

            self.save\_file()

            QCloseEvent.accept()

        elif choice == QMessageBox.No:

            QCloseEvent.ignore()

if \_\_name\_\_ == '\_\_main\_\_':

    app = QApplication(sys.argv)

    myWin = MyWindow()

    myWin.show()

    sys.exit(app.exec\_())

1.4

from Film import film

class linked\_list():

    class node():

        def \_\_init\_\_(self,film):

            self.film = film

            self.next = None

            self.strs=''

    def \_\_init\_\_(self):

        self.head = None

    def \_append(self,film,node):

        if self.head == None:

            self.head = self.node(film)

        elif node.next is not None:

            self.\_append(film,node.next)

        else:

            node.next = self.node(film)

    def append(self,film):

        self.\_append(film,self.head)

    def is\_empty(self):

        if self.head is None:

            return True

        else:

            return False

    def delete(self,film\_name):

        p=self.head

        q=self.head

        if p is not None:

            if p.film.name == film\_name:

                self.head=p.next

                return

            q=q.next

            while q is not None:

                if q.film.name == film\_name:

                    p.next=q.next

                    break

                q=q.next

                p=p.next

    def \_show(self,node):

        if node is not None:

            print(node.film.name,end=' ')

            self.strs=self.strs+'   '+node.film.name

            self.\_show(node.next)

    def show(self):

        self.strs=''

        self.\_show(self.head)

        return self.strs

    def lenth(self):

        i=0

        p=self.head

        while p!=None:

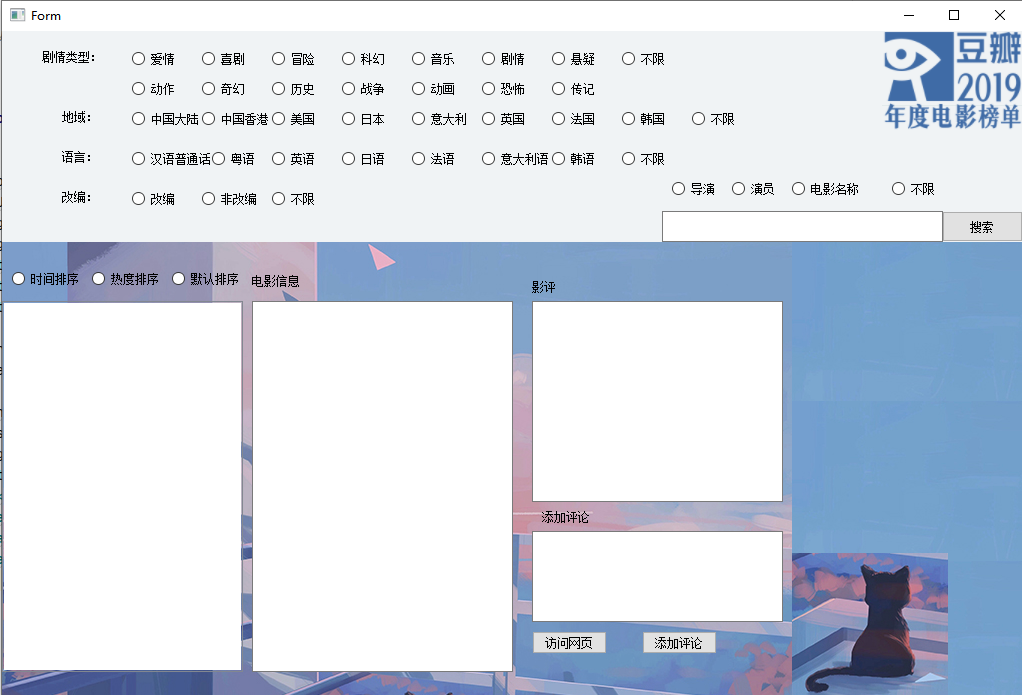
            p=p.next

            i=i+1

        return i

# 

# 4用户手册及测试结果

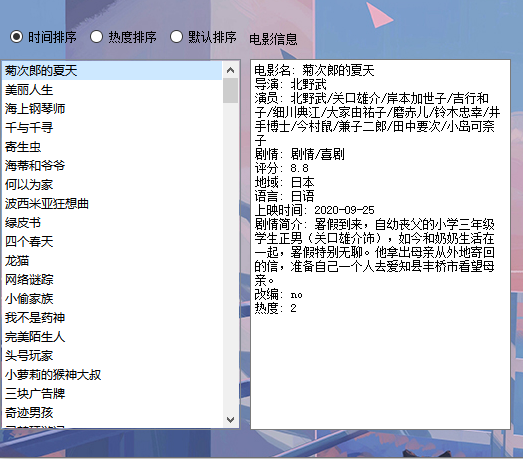


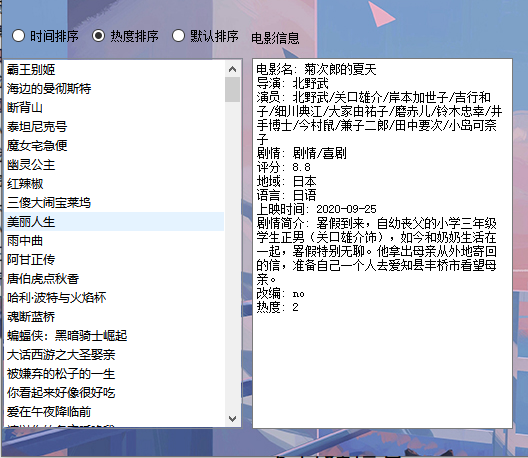
全部选择不限，再点击搜索，可以看到全部250部电影的信息，并可以点击查看详情



## 1、排序演示

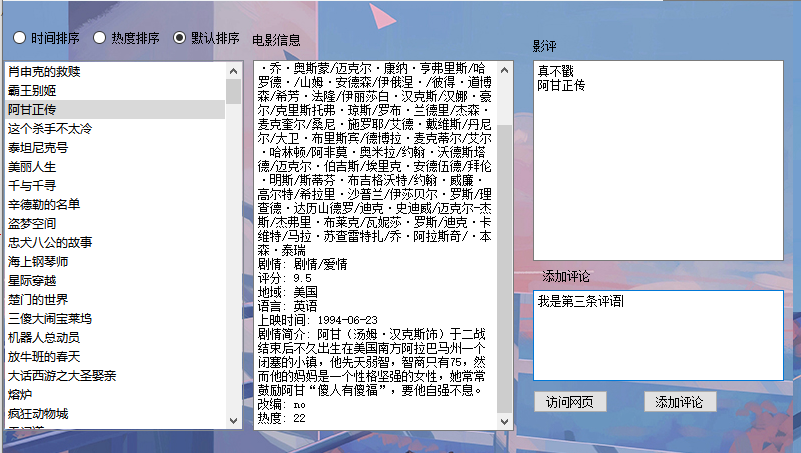
在未选择排序时，默认为默认排序，可以在左侧点击切换排序。



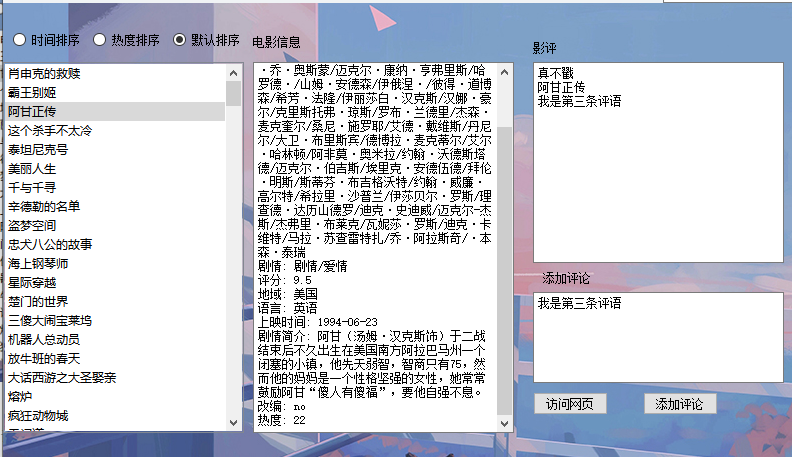


## 2、添加评论

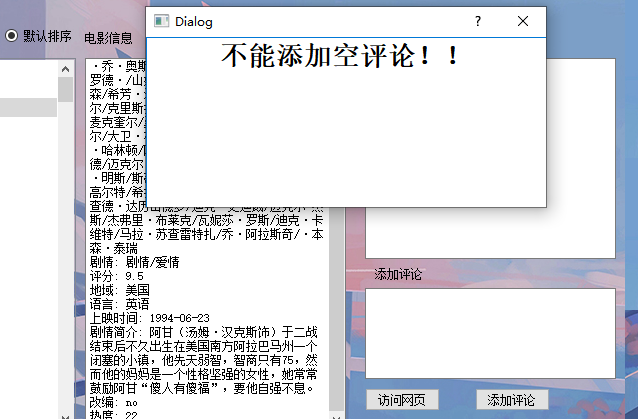
之后可以为阿甘正传添加评语，在右下角添加评论框中输入评语



点击添加，成功显示

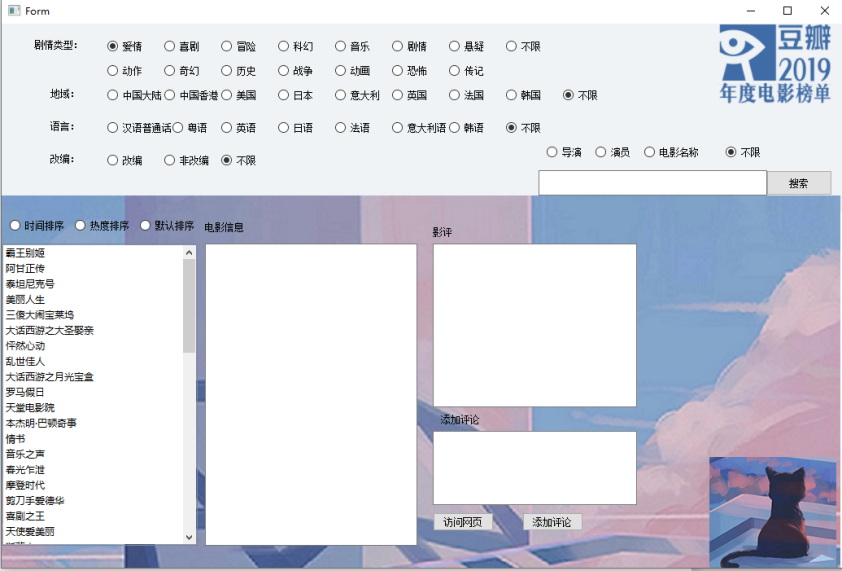


当评论为空时，会提示错误

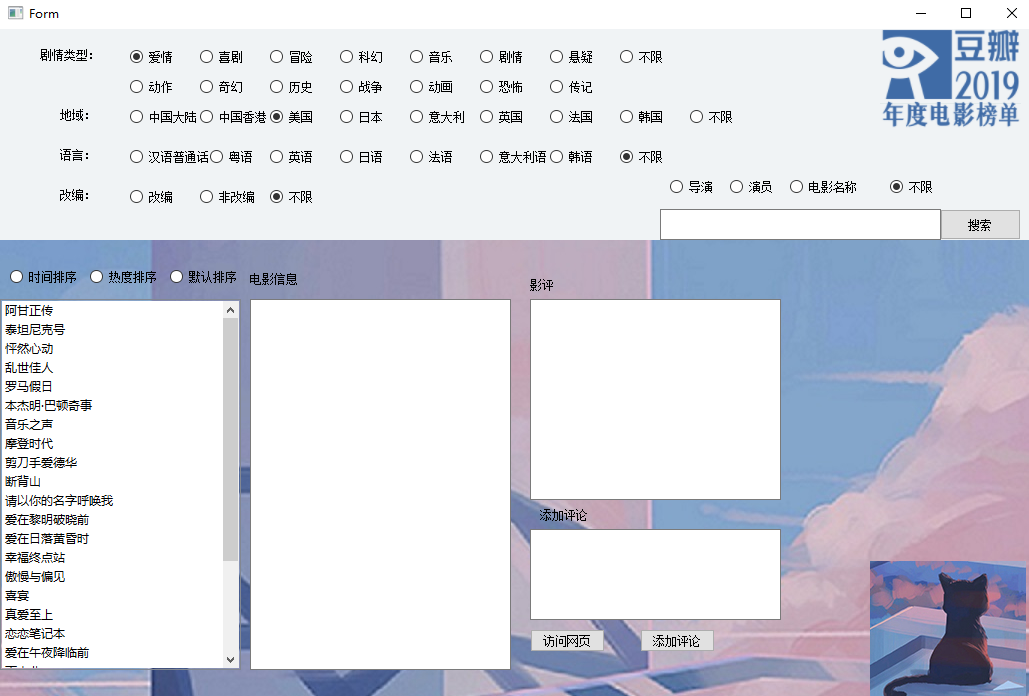


## 交叉查询演示

点击爱情，搜索，在左侧显示了全部爱情类型的电影。



进一步选择美国，语言，非改编，可以看到搜索的范围变小了

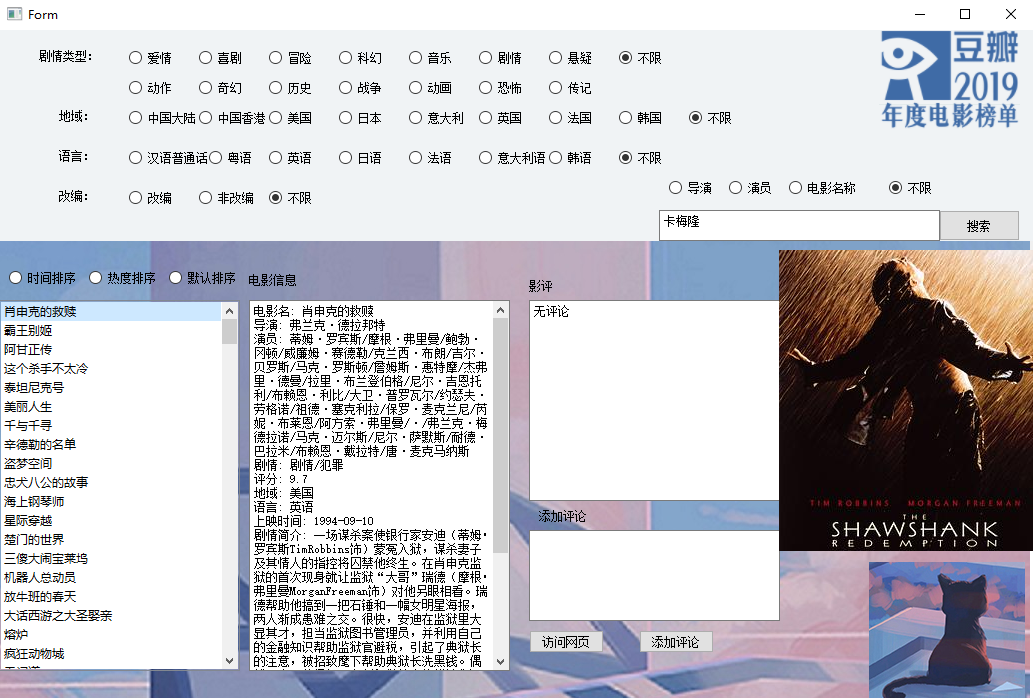


在进一步缩小范围，选择导演，输入卡梅隆，就只剩下泰坦尼克号了



## 访问网页

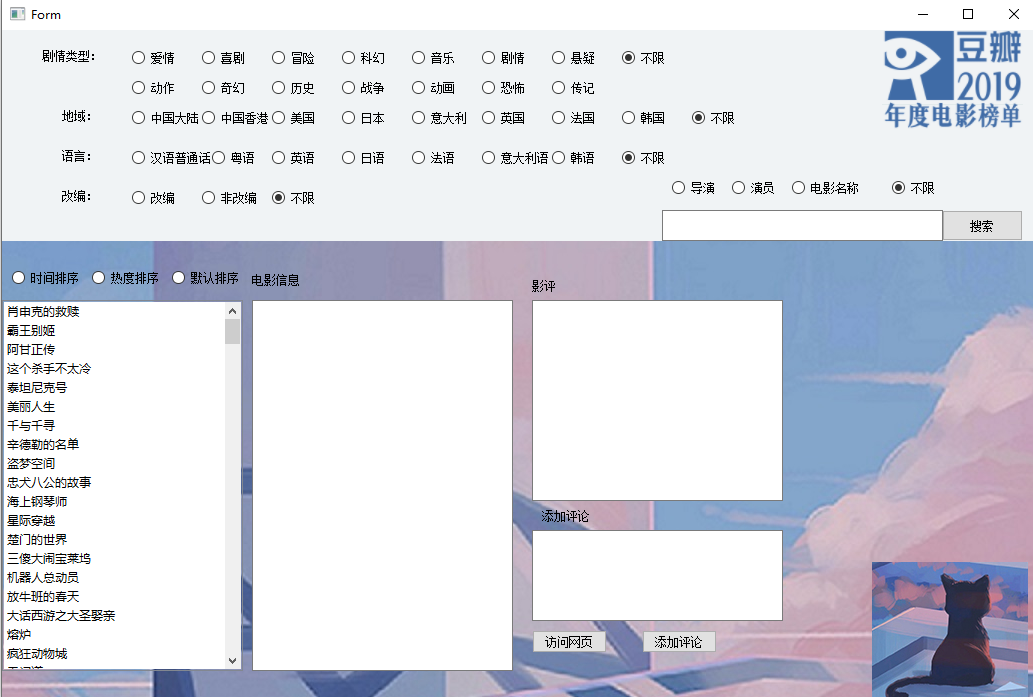
点击访问网页，会自动打开浏览器，弹出该电影的豆瓣网页，可以通过豆瓣观看电影的部分画面（很多电影并没有免费的观看网页，因此统一选择了豆瓣）。



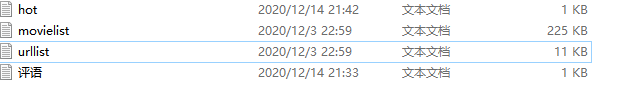


## 5、初始化

全部点击不限，并在此点击搜索，就可以重新搜索（并未找到重置按钮的方法）



## 修改原数据文件





hot.txt 存储电影的热度

urllist.txt 存储电影的网页链接

评语.txt 存储电影的评语

movielist.txt 存储电影的主要数据（不包含热度、网页链接、评语）

热度、网页链接、评语单独存储的目的是方便修改，因为修改数据时是采用全部覆盖的形式，分开存储可以更加效率。

Img文件夹存储图片

# 5总结提高

同学依据自己的经历写出个性化的总结。内容可以包括：程序开发中的体会与收获，开发中遇到的问题与解决情况，自己对自己完成课设情况的评价等等。

## 5.1体会与收获

每次课设的制作都让我收获很多，他能让我重新巩固以前学习的知识，更加熟练的运用编程语言，尤其是这次使用了以前从未使用的python，对于我来说还是有着很大的挑战的。在最开始的阶段，选择哪种数据结构、选择那些算法才能使效率变高让我苦恼不已，这使得我畏手畏脚，迟迟不敢动手，直到后来检查的时间越来越近，才逼迫自己采用最简单、最容易想到的方法先去入手做数据结构的部分，因此选择了思路简单的链表和遍历搜索。

但是这在后面的制作中为我添了很大的麻烦，比如在排序时，链表的快速排序，尤其是单链表的快速排序相比顺序表更复杂、效率也低，此时很多一开始和我思路相同的同学改用了顺序表的存储结构，而我认为通过链表实现快速排序，也是一种练习，因此继续坚持使用了链表，不过还是做出了改进，改为了双向链表。在与其他同学的交流中，我也了解到了树的存储结构的优缺点，可以说通过这次课设，让我对存储结构有了更深刻的认识和体会。

另一个收获就是让我学会了使用python，以前一直听说python是很方便很容易学习理解的语言，并且功能强大，但是一直没有一个时机去学习它，这次课设给我提供了这个时机。虽然在过程中因为不了解python绕了很多弯路（比如python没有指针），但是当到了后期，我真的体会到了python的便利和强大。

最后一个收获就是让我体会到了编程的魅力，以前对编程一直非常抵触，甚至一度因为不想接触编程而想放弃走IT这条路，但是在这次课设中，每当解决一个问题后，所获得的那种巨大的成就感，让我感受到了编程的魅力，这将鼓励我、支持我继续走下去。

## 5.2问题与解决

这次遇到的问题主要有两个方面

一是对python的不熟悉，让我受了很多苦，比如我一开始并不会使用python中的self，再比如没有指针使得我对链表的操作、以及数据更改上遇到了很大的问题，最让我困扰的就是python和c完全是两种存储方式，比如同样的a=1，这两种语言的逻辑完全不同。克服的办法就是在网上搜索解决办法，更深入了解python，实在不行就用些笨办法，比如电影每次重新搜索我都是将数据全部导入一遍。

二是图形界面，python的图形界面用的是pyqt5，学习使用pyqt5和Qtdesiner就花了不少的时间。在制作的过程中有时候因为用错了某个功能，让我不得不推翻从做，总之在制作界面时花费了超过制作功能模块的时间。

## 5.3评价

在界面的美化上也不够精致，应该吧电影图片和电影名字一起在搜索结果中显示。

# 6参考文献

1. 《a byte of python》 Swaroop C H

2、《数据结构与算法》（张铭版）

3、PyQt5，弹窗，自定义输入对话框

<https://blog.csdn.net/qq_39315153/article/details/88241517?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromBaidu-1.control&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromBaidu-1.control>

4、PyQt5-按钮控件使用

<https://www.cnblogs.com/ygzhaof/p/10059540.html>

5、QT designer的安装与汉化（pycharm）

https://blog.csdn.net/qq\_36374896/article/details/83513828