FIN：演员的小世界--实验报告

吴熙楠　　物理学院　　1900011413

一. 实验源代码

1. 队列的实现

```
class Queue:  # 队列
    def __init__(self):
        self.items=[]

    def isEmpty(self):
        return self.items==[]

    def enqueue(self,item):
        self.items.insert(0,item)

    def dequeue(self):
        return self.items.pop()

    def size(self):
        return len(self.items)
```

2. 顶点类的实现

```
class Vertex:  # 顶点
    def __init__(self,key):
        self.name=key  # 演员名字
        self.films=[]  # 出演的电影名字列表
        self.films_id=[]  # 出演的电影id列表
        self.connectedTo={}  # 与该顶点相连的其它顶点为key，权重为value的映射
        self.color='white'
        self.dist=0  # 与起始顶点的距离
        self.pre=None  # 先驱顶点

    def addNeighbor(self,nbr,weight):
        self.connectedTo[nbr]=weight

    def addFilm(self,film,film_id):
        self.films.append(film)
        self.films_id.append(film_id)

    def setColor(self,color):
        self.color=color

    def setDistance(self,d):
        self.dist=d

    def setPre(self,pred):
        self.pre=pred

    def getConnections(self):
        return self.connectedTo.keys()
```

3. 图类的实现

```
class Graph:  # 图
    def __init__(self):#图初始化，属性为顶点数量和key到顶点的映射表
        self.vertices={}
        self.numVertices=0

    def addVertex(self,key):#添加新顶点:return: 新结点

        self.numVertices=self.numVertices+1
        newVertex=Vertex(key)
        self.vertices[key]=newVertex
        return newVertex

    def getVertex(self,key):#通过key查找顶点,return: 若存在key返回key对应的顶点，否则返回None

        if key in self.vertices:
            return self.vertices[key]
        else:
            return None

    def __getitem__(self,item):
        return self.getVertex(item)

    def __contains__(self,v):
        return v in self.vertices
```

```
    def addEdge(self,f,t,weight=1):
        """
        增加图的边
        f: 起始顶点的key值
        t: 终止顶点的key值
        weight: 权重，默认为1
        return: None
        """
        if f not in self.vertices:
            nv=self.addVertex(f)
        if t not in self.vertices:
            nv=self.addVertex(t)
        self.vertices[f].addNeighbor(self.vertices[t],weight)

    def getVertices(self):#返回顶点的key值列表

        return list(self.vertices.keys())

    def __iter__(self):#返回所有顶点的可迭代对象

        return iter(self.vertices.values())
```

4. 连通分支类的实现

```
class Component:#连通分支类
    def __init__(self):
        #生成一个连通分支的实例，属性包括它的规模、演员、电影id、直径等
        self.scale=0#规模，即演员数量
        self.actors=[]#该连通分支内所有演员的列表
        self.films_id=[]#连通分支内所有电影id列表
        self.most_type=[]#连通分支内电影所属类别的前三名
        self.diameter=-1#直径
        self.aver_star=0#电影的平均星级

    def addActor(self,actor):
        #在联通分支内添加演员
        self.actors.append(actor)
        self.scale+=1

    def addFilm(self,film_id):
        #在连通分支内添加电影
        self.films_id.append(film_id)

    def averageStar(self):
        #计算当前连通分支内所有电影的平均星级
        self.aver_star=sum([film_dict[x]["star"] for x in self.films_id]) / len(self.films_id)
```

5. 用 bfs 算法计算当前连通分支的直径

```
    def maxDiameter(self):
        #用bfs算法计算当前连通分支直径
        graph = bulidGraph([film_dict[x] for x in self.films_id])#生成一个连通图
        for v in graph:#对每一个顶点用bfs计算到它的最大距离
            for t in graph:#重置颜色
                t.setColor('white')
            vertex_queue=Queue()
            vertex_queue.enqueue(v)
            v.setColor('gray')#入队后的顶点将color设为'grey'，避免再次入队
            v.setDistance(0)#将起始顶点的距离设为0
            while not vertex_queue.isEmpty():
                temp=vertex_queue.dequeue()
                for nbr in temp.connectedTo:#将相邻未入过队的顶点入队
                    if nbr.color=='white':
                        vertex_queue.enqueue(nbr)
                        nbr.setColor('gray')#入队后的顶点将color设为'grey'，避免再次入队
                        nbr.setDistance(temp.dist+1)#对应距离加1
            self.diameter=max(self.diameter,temp.dist)#取最大距离
```

6. 读取数据

```
import json

f=open('Film.json','rb')
film_data=json.load(f)  # 从json文件中读取电影数据
film_dict={x["_id"]["$oid"]: x for x in film_data}  # 从id到电影列表的映射，便于用id查找电影
```

7. bfs 算法

```
def bfs(graph):
    #找到所有的连通分支,返回一个列表，元素为Component类，列表包含所有的连通分支，并按规模排序，其中仅有一人的分支按演员名字排序
    for v in graph:
        v.setColor('white')
    count=-1#计数
    connected_components=[None] * 5000
    for v in graph:
        if v.color=='white':
            vertex_queue=Queue()
            count += 1#一个新连通分支
            vertex_queue.enqueue(v)
            connected_components[count]=Component()#生成一个Component类
            v.setColor('gray')
            while not vertex_queue.isEmpty():#bfs
                temp=vertex_queue.dequeue()
                connected_components[count].addActor(temp.name)#在Component类中加入演员和电影
                for _id in temp.films_id:
                    connected_components[count].addFilm(_id)
                for nbr in temp.connectedTo:
                    if nbr.color=='white':
                        vertex_queue.enqueue(nbr)
                        nbr.setColor('gray')
    connected_components=connected_components[:count + 1]#去除多余的None
    connected_components.sort(key=lambda x: len(x.actors), reverse=True)#按规模排序
    i = 0
    while i < len(connected_components):#查找第一个仅有一个演员的Component的索引
        if len(connected_components[i].actors)==1:
            break
        i += 1
    connected_components=connected_components[:i]+sorted(connected_components[i:],key=lambda x: x.actors[0])#后面的按名字排序
    return connected_components
```

8. 用电影信息生成图

```
def bulidGraph(filmdata):
    #通过电影信息列表生成一个图
    graph = Graph()
    for film in filmdata:
        title=film["title"]
        _id=film["_id"]["$oid"]
        coactors=[x for x in film["actor"].split(',') ]#一部电影中所有演员的列表
        for a in coactors:
            for b in coactors:
                if a!=b:
                    graph.addEdge(a, b)#添加边，同时生成顶点
        for actor in coactors:#为演员顶点添加电影
            if actor not in graph:
                graph.addVertex(actor)
            graph[actor].addFilm(title, _id)
    return graph
```

9. 查找电影所属类别前三名并返回列表

```python
def topThreeType(films_id_list):
    #查找电影所属类别的前三名,返回前三名的列表
    from collections import Counter
    type_counter=Counter()
    for _id in films_id_list:
        thetype = film_dict[_id]["type"].split(',')
        for t in thetype:
            if t in type_counter:
                type_counter[t]+=1
            else:
                type_counter[t]=1
    typelist=[]
    for x in type_counter.most_common(3):
        typelist.append(x[0])
    return typelist


if __name__ == '__main__':
    actor_graph = bulidGraph(film_data)#生成图

    comp = bfs(actor_graph)
    num = len(comp)#连通分支数

    for c in comp:#计算每个连通分支的类型、直径、平均星级
        c.most_type=topThreeType(c.films_id)
        if c.scale < 100:
            c.maxDiameter()
        c.averageStar()
```

11. 第 1，2 小题

```python
"""
第1、2小题
"""
print(f'联通分支总数：{len(comp)}')
numlist = list(range(20)) + list(range(-20, 0))  # 需要输出的索引：前20个和后20个
print(f"{'序号':<5s}{'演员数':<10s}{'类别前三名':<20s}{'直径':<6s}{'平均星级':<10s}")
def chineseLen(lst):
    return len(''.join(lst))
for i in numlist:#输出结果对齐
    print(
        f"{i:<6d}{comp[i].scale:<8d}{str(comp[i].most_type) + chr(12288) * (7 - chineseLen(comp[i].most_type)):<24s}{comp[i].diameter:<6d}
"""
```

12. 第 3 小题

```python
"""
第3小题
"""
import matplotlib.pyplot as plt
from matplotlib.gridspec import GridSpec
from brokenaxes import brokenaxes
sps1, sps2, sps3=GridSpec(3,1)
plt.figure(figsize=(12, 12))
# 分割
bax=brokenaxes(ylims=((0,50), (84675, 84700)),subplot_spec=sps1)
bax.set_title('size')
bax.bar(range(40),[x.scale for x in comp[:20]] + [x.scale for x in comp[-20:]],tick_label=numlist)
plt.subplot(312)
plt.bar(range(40),[17] + [x.diameter for x in comp[1:20]] + [x.diameter for x in comp[-20:]],tick_label=numlist)
plt.title('diameter')
plt.subplot(313)
plt.bar(range(40),[x.aver_star for x in comp[:20]] + [x.aver_star for x in comp[-20:]],tick_label=numlist)
plt.title('average star')

plt.show()

"""
```

13. 第 4 小题

```
"""
第4小题
"""
zhou = actor_graph['周星驰']
coactor = ['周星驰'] + [x.name for x in zhou.connectedTo]
all_films = set()
for actor in coactor:
    all_films = all_films | set(actor_graph[actor].films_id)

print(f"周星驰电影平均星级：{sum([film_dict[x]['star'] for x in zhou.films_id]) / len(zhou.films_id):.2f}")
```

14. 第 5 小题

```
"""
第5小题
"""
print(f"周星驰和他的共同出演者，有：{len(coactor)}人")
print(f"他们各自一共出演了电影：{len(all_films)}部")
print(f"所出演的电影平均星级：{sum([film_dict[x]['star'] for x in all_films]) / len(all_films):.2f}")
print(f"电影所属类别的前三名：{topThreeType(all_films)}")
```

15. 第 6 小题（看看我最喜欢的两位声优出演电影情况）

```
"""
第6小题
"""
a = actor_graph['花泽香菜']
coactor1 = ['花泽香菜'] + [x.name for x in a.connectedTo]
all_films1 = set()
for actor in coactor1:
    all_films1 = all_films1 | set(actor_graph[actor].films_id)
b = actor_graph['日高里菜']
coactor2 = ['日高里菜'] + [x.name for x in a.connectedTo]
all_films2 = set()
for actor in coactor1:
    all_films2 = all_films2 | set(actor_graph[actor].films_id)
print(f"花泽香菜电影平均星级：{sum([film_dict[x]['star'] for x in a.films_id]) / len(a.films_id):.2f}")
print(f"花泽香菜所出演电影：{a.films}")
print(f"花泽香菜电影类别前三:{topThreeType(all_films1)}")
print(f"日高里菜电影平均星级：{sum([film_dict[x]['star'] for x in b.films_id]) / len(b.films_id):.2f}")
print(f"日高里菜所出演电影：{b.films}")
print(f"日高里菜电影类别前三:{topThreeType(all_films2)}")
```

二. 数据分析

本次的几万条数据中，有一些比如由于部分数据的特殊而产生的空演员，并且这些电影中有重复的电影名字，这些都是会影响运行结果的，但通过结果来看，似乎只会对于最大联通分支产生影响，并且由于数据量庞大，这些问题带来的影响几乎可以忽略（个位数影响）。

三. 运行结果说明

1. 第 1 小题

通过第 1 小题的运行结果，我们可以看到在我们所构建的整个由电影构成的图内，联通分支总数为 4577 个；但是，这个结果是我们在添加电影时未去除空串的结果，如果去除空串后，联通分支总数将变为 4583 个。

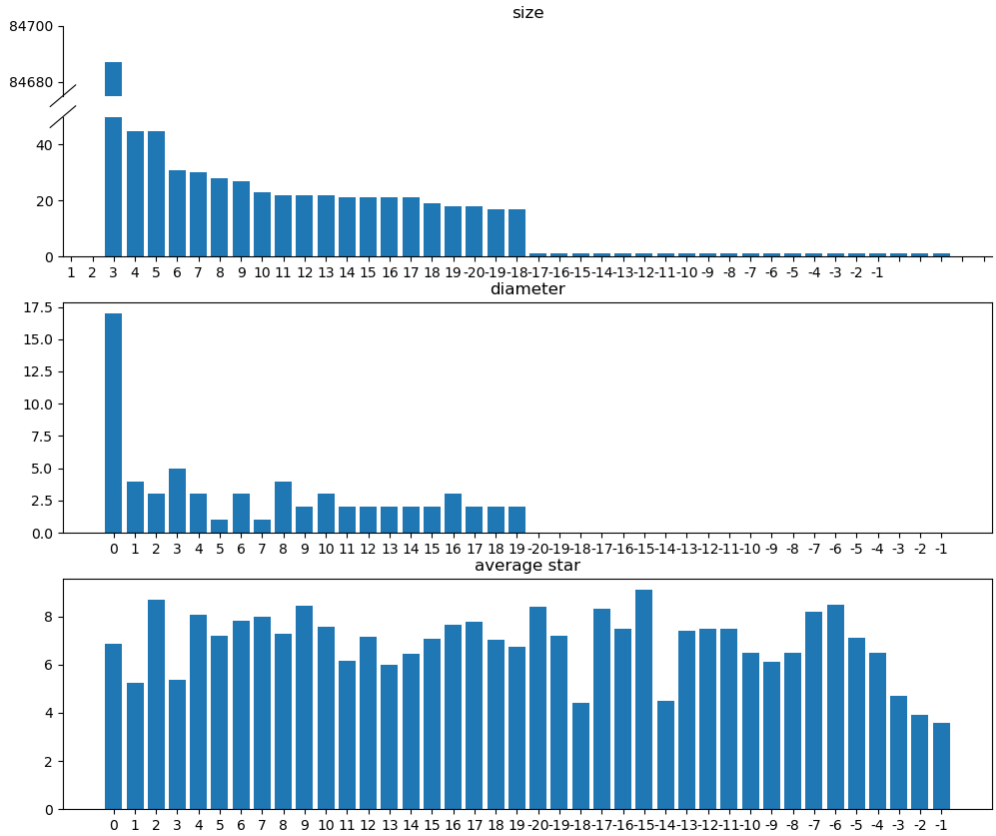
2. 第 2 小题

序号	演员数	类别前三名	直径	平均星级
0	84687	['剧情', '喜剧', '动作']	-1	6.88
1	45	['恐怖', '历史', '战争']	4	5.24
2	45	['纪录片', '历史', '战争']	3	8.67
3	31	['剧情', '犯罪', '同性']	5	5.37
4	30	['剧情', '爱情']	3	8.06
5	28	['剧情', '动作', '历史']	1	7.20
6	27	['剧情', '喜剧', '恐怖']	3	7.81
7	23	['剧情', '音乐']	1	8.00
8	22	['剧情', '爱情', '犯罪']	4	7.30
9	22	['剧情', '喜剧', '儿童']	2	8.43
10	22	['剧情', '悬疑']	3	7.57
11	21	['剧情', '传记', '历史']	2	6.14
12	21	['剧情', '喜剧']	2	7.17
13	21	['情色', '剧情', '爱情']	2	6.00
14	21	['历史', '战争', '剧情']	2	6.43
15	19	['剧情', '同性', '爱情']	2	7.08
16	18	['剧情', '喜剧', '同性']	3	7.66
17	18	['剧情']	2	7.78
18	17	['剧情', '喜剧', '同性']	2	7.03
19	17	['剧情', '科幻', '悬疑']	2	6.74
-20	1	['剧情']	0	8.40
-19	1	['剧情', '同性']	0	7.20
-18	1	['剧情', '喜剧', '恐怖']	0	4.40
-17	1	['剧情', '运动']	0	8.30
-16	1	['剧情']	0	7.50
-15	1	['剧情', '传记']	0	9.10
-14	1	['剧情', '悬疑', '惊悚']	0	4.50
-13	1	['动画', '短片', '儿童']	0	7.40
-12	1	['剧情']	0	7.50
-11	1	['喜剧']	0	7.50
-10	1	['恐怖']	0	6.50
-9	1	['剧情', '喜剧']	0	6.10
-8	1	['剧情', '喜剧', '动画']	0	6.50
-7	1	['纪录片', '同性']	0	8.20
-6	1	['武侠', '古装', '剧情']	0	8.50
-5	1	['惊悚', '纪录片']	0	7.10
-4	1	['剧情', '惊悚', '奇幻']	0	6.50
-3	1	['剧情', '惊悚']	0	4.70
-2	1	['剧情', '情色']	0	3.90
-1	1	['爱情', '情色']	0	3.60

通过运行结果我们可以看到，规模最大的联通分支演员数达到了 84687 个，同样，这个数据是未去除空串的结果，去除后的结果变为 84671 个演员，其余联通分支不变。序号 0-19 代表着规模前二十的联通分支，序号（-1）-（-20）代表着规模倒数前二十的联通分支。其中，最大联通分支类别前三名为“剧情”“喜剧”“动作”，直径由于

过大，故直接取为-1 不作计算，电影平均星级为 6.88，说明电影良莠不齐。而其余联通分支的直径，电影平均星级等等不做一一列举。

3. 第 3 小题



通过运行结果我们可以看到，在联通分支的规模上，除开第一个联通分支外，其余联通分支之间的规模差距都不大，后 20 名的规模均为 1 个演员；在联通分支的直径上，除开第一个联通分支直径直接设为 17 以外，其余联通分支的直径差别并不大，后 20 名的直径均为 0；在平均星级方面，我们发现整体平均星级差别不大，但低星级多出现于联通分支规模较小的一方。

4. 第 4，5 小题

```
周星驰电影平均星级：7.19
周星驰和他的共同出演者，有：302人
他们各自一共出演了电影：3132部
所出演的电影平均星级：6.26
电影所属类别的前三名：['动作', '剧情', '喜剧']
```

通过运行结果我们可以发现，周星驰电影平均星级为 7.19；周星驰和他的共同出演者一共有 302 人，他们各自一共出演了电影 3132 部；所出演的电影平均星级：6.26，电影所属类别的前三名分别为“动作”，“剧情”，“喜剧”。

5. 第 6 小题

```
花泽香菜电影平均星级：7.39
花泽香菜所出演电影：['魔法科高校的劣等生 呼唤星辰的少女 劇場版 魔法科高校の劣等生 星を呼ぶ少女', '尸者帝国 屍者の帝国', '黑岩射手 ブラック★ロックシューター', '哥斯拉：怪兽行星 GODZILLA 怪獣惑星', '剧场版 破刃之剑 第一章 觉醒之刻 劇場版 ブレイク ブレイド 第一章 覚醒ノ刻', '伪恋OAD：新婚/魔法甜点师小咲酱!! ニセコイ シンコン/マジカルパティシエ小咲ちゃん!!', '伪恋OAD：澡堂/福利 ニセコイ セントウ/サービス', '我们大家的河合庄OVA 僕らはみんな河合荘 初めての', '伪恋OAD：兼职/酒后吐真言 ニセコイ ヘンボウ/オシゴト', '伪恋OAD：丢失/巫女小姐 ニセコイ フンシツ/ミコサン', '变态生理研讨会 OAD 変ゼミ OAD', '监狱学园OAD：疯狂的蜡 監獄学園 マッドワックス', '春宵苦短，少女前进吧！ 夜は短し歩けよ乙女', '游戏王：次元的黑暗面 遊☆戯☆王 THE DARK SIDE OF DIMENSIONS', '千子 センコロール', '你的名字。 君の名は。', '命运之夜原型 Fate/Prototype', 'AURA～魔龙院光牙最后的战斗～ AURA ～魔竜院光牙最後の闘い～', '属性同好会 OAD ディーふらぐ! OAD ウォーター!!', '我的朋友很少 Add-on Disk 僕は友達が少ない あどおんでぃすく', '勇猛宇宙海贼：亚空间之深渊 モーレツ宇宙海賊 ABYSS OF HYPERSPACE -亜空の深淵-', '妖狐×仆SS TV未放送第13话 妖狐×僕SS TV未放送の第13話', '上课小动作OAD：倒棒游戏&猫 となりの関くん 「棒倒し/猫」', '橘色奇迹 -未来- orange -未来-', '言叶之庭 言の葉の庭', '烟花 打ち上げ花火、下から見るか？横から見るか？', '重神机潘多拉 重神機パンドーラ', '机动战士高达AGE 機動戦士ガンダムAGE', '无限斯特拉托斯2 IS＜インフィニット・ストラトス＞2', 'Rewrite リライト', '核爆默示录 COPPELION', '恋爱禁止的世界 恋と嘘', '无限斯特拉托斯OVA：一夏的思绪 IS＜インフィニット・ストラトス＞ 一夏の想いで', '无限斯特拉托斯：世界清洗篇 IS＜インフィニット・ストラトス＞ ワールド・パージ編', '破刃之剑 (TV版) ブレイクブレイド', '零度战姬Vibration フリージング ヴァイブレーション', '玩伴猫耳娘 あそびにいくヨ！', 'Buddy Complex バディ・コンプレックス', '无限斯特拉托斯 Encore 渴望恋爱的六重奏 IS＜インフィニット・ストラトス＞ アンコール「恋に焦がれる六重奏」', '虫之歌 ムシウタ', '境界触发者 ワールドトリガー', '无限斯特拉托斯 IS＜インフィニット・ストラトス＞', '青之驱魔师 京都不浄王篇：蛇与毒 青の祓魔師 京都不浄王編第拾参話 OAD 蛇と毒', '分形世界 フラクタル', '橘色奇迹 orange', '创圣大天使EVOL アクエリオンEVOL', 'Rewrite 第二季 Moon篇/Terra篇 Rewrite 2nd シーズン Moon編 / Terra編', '剧场版 破刃之剑 第二章 诀别之路 劇場版 ブレイク ブレイド 第二章 訣別ノ路', '机动战士高达AGE 伊甸的回忆 機動戦士ガンダムAGE Memory of Eden', '机械篮球 バスカッシュ！', '暴力宇宙海贼 モーレツ宇宙海賊', '剧场版 破刃之剑 第六章 恸哭之塞 劇場版 ブレイク ブレイド 第六章 慟哭ノ砦', '世纪末超自然学院 世紀末オカルト学院', '绝园的暴风雨 絶園のテンペスト', '罪恶王冠 ギルティクラウン', '.hack//Quantum', '剧场版 破刃之剑 第三章 凶刃之痕 劇場版 ブレイク ブレイド 第三章 凶刃ノ痕', '苍色骑士 ブラスレイター', '古城荆棘王 いばらの王', '黑之契约者2：流星的双子 DARKER THAN BLACK -流星の双子-', 'BLAME!', '命运石之门：聪明睿智的认知计算 STEINS;GATE 聡明叡智のコグニティブ・コンピューティング', '剧场版 破刃之剑 第四章 惨祸之地 劇場版 ブレイク ブレイド 第四章 惨禍ノ地', '剧场版 破刃之剑 第五章 死线之涯 劇場版 ブレイク ブレイド 第五章 死線ノ涯', '某科学的超电磁炮S とある科学の超電磁砲S', '出包王女Darkness OAD6 To LOVEる -とらぶる- ダークネス OAD6', '出包王女Darkness OAD3 To LOVEる -とらぶる- ダークネス OAD3', '来自新世界 新世界より', '寄生兽 生命的准则 寄生獣 セイの格率', '出包王女Darkness OAD9 To LOVEる -とらぶる- ダークネス OAD9', '命运石之门23β：境界面上的缺失之环 STEINS;GATE #23(β) 境界面上のミッシングリンク', '出包王女Darkness OAD5 To LOVEる -とらぶる- ダークネス OAD5', '命运石之门：横行跋扈的浪荡之徒 STEINS;GATE 横行跋扈のポリオマニア', '命运石之门剧场版：负荷领域的既视感 劇場版 STEINS;GATE 負荷領域のデジャヴ', '重启咲良田 サクラダリセット', '少女终末旅行 少女終末旅行', '命运石之门 STEINS;GATE', '命运石之门0 STEINS;GATE 0', '弹丸论破3 -The End of 希望之峰学园- 希望篇 ダンガンロンパ3 -The End of 希望ヶ峰学園- 希望編', '无头骑士异闻录：天网恢恢 デュラララ!! 第12.5話 天網恢恢']
花泽香菜电影类别前三:['动画', '剧情', '科幻']

日高里菜电影平均星级：7.28
日高里菜所出演电影：['终结的炽天使OVA：吸血鬼夏哈尔 終わりのセラフ 吸血鬼シャハル', '刀剑神域：序列之争 劇場版 ソードアート・オンライン -オーディナル・スケール', '追逐繁星的孩子 星を追う子ども', '四月一日灵异事件簿：春梦记 前篇 xxxHOLiC 春夢記 前編', '妖狐×仆SS TV未放送第13话 妖狐×僕SS TV未放送の第13話', '四月一日灵异事件簿：春梦记 后篇 xxxHOLiC 春夢記 後編', '摇滚都市 第二季 SHOW BY ROCK!!#', '摇滚都市 第一季 SHOW BY ROCK!!', '漆黑的子弹 ブラック・ブレット', '地球队长 キャプテン・アース', '刀剑神域 Extra Edition ソードアート・オンライン Extra Edition', '伽利略少女 ガリレイドンナ', '舰队收藏 艦隊これくしょん -艦これ-', '苍蓝钢铁的琶音 劇場版 劇場版 蒼き鋼のアルペジオ-アルス・ノヴァ-DC', '刀剑神域2 ソードアート・オンラインⅡ', '银河美少年 劇場版 スタードライバー THE MOVIE', '在盛夏等待 あの夏で待ってる', '刀剑神域 ソードアート・オンライン', '苍蓝钢铁的琶音剧场版：Cadenza 劇場版 蒼き鋼のアルペジオ -アルス・ノヴァ- Cadenza', '出包王女Darkness OAD9 To LOVEる -とらぶる- ダークネス OAD9', '出包王女Darkness OAD7 To LOVEる -とらぶる- ダークネス OAD7']
日高里菜电影类别前三:['动画', '剧情', '科幻']
```

现在二次元文化在中国算是比较普遍的一种现象，因此对于喜欢的日本的声优我们也有必要了解其出演电影情况，而这次作业显然给了我们一个绝佳的机会，从上述数据我们可以看到众多日本动漫平均星级都是 7-8 之间，而电影类别最多为剧情和科幻；花泽香菜相对于日高里菜较早出道，因此出演电影也较多。