2021-2022 学年秋季学期

《创新创业实训（1）》(0869A002)

课程报告

成绩	

学　　号	20123101	学　　院	计算机工程与科学学院
姓　　名	李昀哲	手工签名	
报告题目	**逐步求精算法设计及爬虫的理解与应用**		

实训报告成绩 50%	实训过程描述：清晰、全面。（5%）	
	报告主要部分：1、思维实例叙述清楚、有意义；2、分析到位；3、思路独特或有创意；4、关键实现技术描述清楚详细；5、内容丰富；6、不直接粘贴所有代码；7、代码有注释或说明。（25%）	
	实训收获体会：感受真实、深刻，建议有价值。（10%）	
	书写格式：书写规范、用词正确、无明显的错别字，图表、代码清晰规范，格式协调、效果好，参考文献书写、引用规范合理。（10%）	
工作实绩 50%	1、平时表现、进步情况（25%），2、工作实绩，项目验收（25%）	

教师对该生工作实绩简要评述：

教师签名：

日期：　　年　　月　　日

逐步求精算法设计及爬虫的理解与应用

20123101 李昀哲

计算机工程与科学学院

1 本学期实训过程简述

本学期的实训内容从两方面开展：首先是对于“逐步求精，自顶向下”编程思想的学习，并将这种思想应用于“八皇后问题”的算法设计和实现中，培养了高效、准确的编程思想，为今后的学习打下坚实基础；其次，是对于 Python 爬虫的学习与应用，针对导师提供的资料和上网查阅，逐步了解 Python 爬虫的步骤和必要“组成元素”。根据上述资料，先简单从零开始实现了简单的网页爬虫，记录过程中遇到的问题和解决办法，加深了对爬虫的理解；之后利用提供的代码资料加以优化，应用于“Machine Learning”，“Material”两个关键词在 Web of Science 论文网站上的爬取，并整理为表格。

2 所从事课题涉及的典型问题与关键技术实现

1.1 “逐步求精”算法设计解决“八皇后问题”

1.1.1 问题描述

在 8 x 8 的国际象棋棋盘上有 8 个皇后，将它摆成如下格局：使谁也吃不掉谁。即每一行、每一列和每一对角线上最多只有一个皇后。要求构造一个程序，产生所有的这种格局。

1.1.2 基于逐步求精的算法设计思路

需要得到满足要求的格局集合，这个集合一定是全部格局的子集。不妨设所要求的集合为 ***P***，所有格局的全集为 ***U***。

最基本的思路为：产生一个元素 u_i，判断这个元素是否满足要求，即是否为 ***P*** 的子集，是，则打印输出；否，则产生下一个 u_i。流程图如**图 1.程序流程实现**所示。接着对流程图中的各部分进行求精。

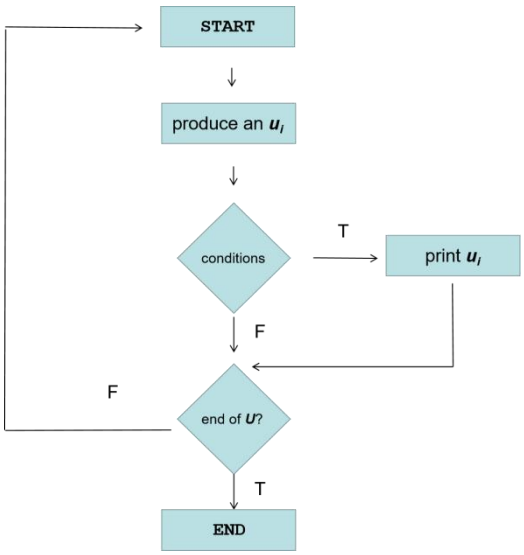

图 1. 程序流程实现

首先确定全集 ***U*** 来说，由于每行、列只含有一个皇后，且棋盘大小为 8 x 8，所以使用循环，每次产生一个元素，产生 8 个后退出循环。

```
//block A
   N = 8;
   i = 1;
   While (i <= N)
   {
       "产生下一个元素";
        i ++;
   }
```

接下来对“produce next element”进行求精，棋盘中的行从上至下 1-8 进行编号，列从左至右进行编号，如**图 2.对棋盘的编号示例（灰色表示假定第一个皇后的位置）**，每次产生的元素为 a[i] = k，表示第 i 行的 k 列放置皇后。每次产生一个放置在[1, k]的皇后后续满足条件的所有格局。每一行放置一个，就自动向下移动一行，即 i++。

由此，每一个满足条件的格局最后会输出为 a[1]a[2]...a[8]。

1								
2								
3								
4								
5								
6								
7								
8								
	1	2	3	4	5	6	7	8

图 2. 对棋盘的编号示例

根据上述求精过程，将 while 循环优化为 for 循环达到产生一个格局后移动第一行皇后位置的目的，

“产生下一个元素”代码表示为：

```
//block B
   for (int i = 1; i <= N; i++)
          {
              a[k] = i;
               “产生当第一个皇后位于[a,k]时所有的结果”
          }
```

下一步，对语句“产生当第一个皇后位于[a,k]时所有的结果”进一步求精。做法和前一步类似，当第一个皇后位置确定后，第二个皇后的位置就有了限制，因此要加以判断：

```
//block C
   if ("第 k 行皇后的放置是否正确")
       "放置下一个皇后"
```

判断之后，如果位置正确，就再次重复“产生下一个元素”，直至摆放至第 8 行。程序将会包含 8 个相同的循环，但进

一步求精会发现，这 8 个循环的结构类似，执行语句略有不同（即：第一个摆放的皇后无需验证其合法性）。但对于第一个皇后的放置，添加判断语句也不会出现错误，因此，8 个循环可以用递归函数来实现。block B 就可以改为：

```
//block D
   for (int i = 1; i <= N; i++)
          {
              a[k] = i;
              if (“第 k 行皇后的放置是否正确”)
                  k += 1;
                  “block B”
          }
```

至此，主体程序框架基本完成，对于判断条件“第 k 行皇后的放置是否正确”编写函数实现以提高可读性和重用性：

```
   bool check(int a[], int k)  //传入的值为 queen 的位置
   {
      for (int i = 1; i <= k - 1; i++)
      {
          if ((abs(a[i] - a[k]) == k - i) //判断和之前所有 queen 同时出现在对角的情况
               || (a[i] == a[k]))          //判断和之前所有 queen 同时出现在同一列的情况
              return false;
      }
      return true;
   }
```

同时，将 block D 改写为函数，利于递归调用：

```
   void step_next(int k)             //初始值为 1
   {
      for (int i = 1; i <= N; i++)
      {
          a[k] = i;
          if (check(a, k) == 1)     //判断条件
          {
              step_next(k + 1);     //递归
          }
      }
    }
```

添加输出语句便于查看结果：

```
   void step_next(int k)
   {
      if (k>N)                          //当超过第 8 行时，输出结果
      {
          for (int i = 1; i <= 8; i++)
          {
              std::cout << a[i];
```

```
        }
        std::cout << '\t';
        num++;
        if (num % 5 == 0){std::cout << std::endl;}
    }
    else
    {
        for (int i = 1; i <= N; i++)
        {
            a[k] = i;
            if (check(a, k) == 1)       //判断是否合法
            {
                step_next(k + 1);       //下一步递归
            }
        }
    }
```

主函数中的调用为:

```
int main()
{
    int n = 8, num = 0;
    step_next(1);
    std::cout <<"The ways of queens is " <<num << std::endl;
    return 0;
}
```

输出结果为：（共 92 种解法，每一种的位置如数字所示）

例：15863724 表示为，如**图 3.输出示例**所示

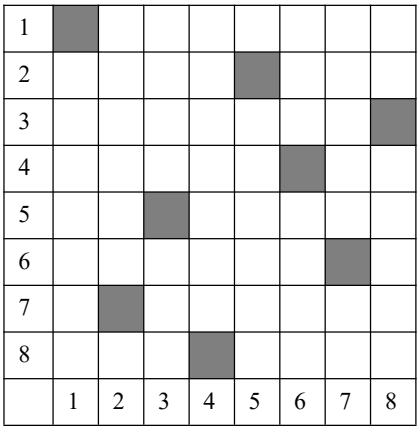

图 3. 输出示例

15863724	16837425	17468253	17582463	24683175
25713864	25741863	26174835	26831475	27368514
27581463	28613574	31758246	35281746	35286471
35714286	35841726	36258174	36271485	36275184

```
36418572  36428571  36814752  36815724  36824175
37285146  37286415  38471625  41582736  41586372
42586137  42736815  42736851  42751863  42857136
42861357  46152837  46827135  46831752  47185263
47382516  47526138  47531682  48136275  48157263
48531726  51468273  51842736  51863724  52468317
52473861  52617483  52814736  53168247  53172864
53847162  57138642  57142863  57248136  57263148
57263184  57413862  58413627  58417263  61528374
62713584  62714853  63175824  63184275  63185247
63571428  63581427  63724815  63728514  63741825
64158273  64285713  64713528  64718253  68241753
71386425  72418536  72631485  73168524  73825164
74258136  74286135  75316824  82417536  82531746
83162574  84136275
The ways of queens is 92
```

1.1.3 设计感悟

逐步求精的程序设计让我学会先整体后局部的设计方法，从大体框架种入手，一边设计一边验证，逐步完善结构细节，这样设计不仅结构清晰，而且可读性极强，对于程序的二次修改和优化带来极大便利。将一个大的算法问题，细化为“如何放置皇后”，“如何验证合法性”，“如何逐层递归”，“如何输出”等子问题，使得问题的解决过程不再枯燥，将各部分完成后进行拼接整合，这也符合当代社会强调各学科间的相互依赖实现科技进步的思想。

1.2 Python爬虫从零开始实现及在论文数据获取中的应用

1.2.1 爬虫从零开始实现

爬虫是模拟浏览器发送请求，一种按照一定的规则，自动地抓取互联网信息的程序。现在信息的获取相对于过去信息面变广了，但无用的信息也变多了，因此筛选是相当重要的。

对于当下前沿的机器学习来说，需要将知识转化为机器能懂的符号语言来进行训练，但知识如上所说，是存在冗余的，真正有价值的数据集是很有限的，那么如何精确、批量的获取我们需要的数据集，就成了机器学习领域除算法外另一个关键点。“爬虫”就是这样一种技术，在大量数据中根据我们指定的要求，分析删选出真正需要的数据。

1.2.1.1 简单爬虫实现步骤

1.确定需要爬取的 url 地址

2.由请求模块向 url 发起请求，并得到网站回应

3.从相应内容中提取数据：

1.所需数据保存

2.页面中有其他跟进的 url，继续第二步去发起请求，如此循环。

```
import urllib.request

#向 url 发送请求返回对象，构建 response 类
response = urllib.request.urlopen('http://www.baidu.com/')
```

```
#response 类的 read()方法，提取响应内容
html = response.read().decode('utf-8')

#显示响应内容
print(html)
```

1.2.1.2 常用的方法

1.urllib 库用以实现 request 的发送

- urllib.request.urlopen()

• 向网站发出请求并获得返回对象

• urllib.request 提供最基本的构造 HTTP 请求的方法，模拟浏览器的一个请求发起

• parameter：网址，等待超时时间

2.响应对象的方法

```
html = response.read().decode() #没有.decode()得到的是 bytes 类型
url = response.geturl()
code = response.getcode() #返回 http 响应的状态码
```

1.2.1.3 网站对于爬虫和人类访问的判断

请求头 User-Agent 就是网站识别我们的信息。

浏览器访问时的 User-Agent 为：

```
"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.54 Safari/537.36 Edg/95.0.1020.30"
```

使用 python 访问：

```
import urllib.request

url = 'http://httpbin.org/get'
response = request.urlopen(url)          #构建响应对象
html = response.read().decode('utf8')    #解码
print(html)
```

得到的 User-Agent：

```
 "User-Agent": "Python-urllib/3.7"
```

由上述可见，通过对“请求头”的辨别，很容易区分人类和爬虫，因此需要对 User Agent 进行重构。

1.2.1.4 重构 User-Agent

```
import urllib.request

url = 'http://httpbin.org/get'
#定义新的 User_Agent
headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.1 (KHTML, like Gecko) Chrome/14.0.835.163 Safari/535.1'}

#Request 函数重构“头”
```

```
req = urllib.request.Request(url=url, headers=headers)
#再次发送请求
res = urllib.request.urlopen(req)
html = res.read().decode()
print(html)
```

得到的 User-Agent：

```
"User-Agent": "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.1 (KHTML, like Gecko)
Chrome/14.0.835.163 Safari/535.1"
```

1.2.1.5 url 编码模块

百度“上海大学”，得到冗长 url：

```
https://www.baidu.com/s?wd=%E4%B8%8A%E6%B5%B7%E5%A4%A7%E5%AD%A6&rsv_spt=1&rsv_iqid=0xf403
3ad4000ac47f&issp=1&f=8&rsv_bp=1&rsv_idx=2&ie=utf-8&tn=15007414_8_dg&rsv_enter=1&rsv_dl=t
b&rsv_sug3=12&rsv_sug1=18&rsv_sug7=101&rsv_sug2=0&rsv_btype=i&inputT=5207&rsv_sug4=5207
```

其中只有如下部分是有用的：

```
    #wd 之后的是上海大学的编码，汉字是需要编码的
    https://www.baidu.com/s?wd=%E4%B8%8A%E6%B5%B7%E5%A4%A7%E5%AD%A6
```

方法及常用参数：

```
import urllib.parse
from urllib import parse

#查询参数{'wd': '上海大学'}
query_string = {'wd': '上海大学'}
result = urllib.parse.urlencode(query_string)
print(result)
#wd=%E4%B8%8A%E6%B5%B7%E5%A4%A7%E5%AD%A6
```

多个参数时

```
#查询多个参数
#百度一个页面显示 10 个信息（除广告外），pn=10 表示第二页
param = {'wd': '上海大学', 'pn': '10'}
result = parse.urlencode(param)
print(result)
#wd=%E4%B8%8A%E6%B5%B7%E5%A4%A7%E5%AD%A6&pn=10
```

由上：可以通过字符串拼接得到需要的 url：（得到百度搜索上海大学的第二页）

```
url = 'http://baidu.com/s?{}'.format(result)
print(url)
#http://baidu.com/s?wd=%E4%B8%8A%E6%B5%B7%E5%A4%A7%E5%AD%A6&pn=10
```

1.2.1.6 拼接 URL 的三种方式

先得到参数编码：

```
params = {'wd':'上海大学', 'pn':'10'}
result = urllib.parse.urlencode(params)
print(result)
```

• 字符串相加

```
baseurl = 'http://baidu.com/s?'
#由 result 得到
param = 'wd=%E4%B8%8A%E6%B5%B7%E5%A4%A7%E5%AD%A6&pn=10'
url = baseurl + param
print(url)
```

• 字符串占位符

```
param = 'wd=%E4%B8%8A%E6%B5%B7%E5%A4%A7%E5%AD%A6&pn=10'
url = 'http://baidu.com/s?%s'%param
print(url)
```

• format：注意要在 baseurl 最后加{}

```
param = 'wd=%E4%B8%8A%E6%B5%B7%E5%A4%A7%E5%AD%A6&pn=10'
baseurl = 'http://baidu.com/s?{}'
url = baseurl.format(param)
print(url)
```

1.2.1.7 技术小结

```
urllib.request
req=request.Request(url=url,headers=headers)
res=request.urlopen(req)
html=res.read().decode()
#响应对象方法
res.read()
res.getcode()
res.geturl()
#urllib.parse
parse.urlencode({params})
parse.quote() #用于解码
parse.unquote()
```

1.2.2 在论文数据获取中的应用

在上述步骤的基础上理解导师所提供的代码。

在 Main_Methods.py 文件中，首先构建 SpiderMain 类，在 init 函数中处理 User-Agent、查询年月等和形成数据相关的参数。

再定义多个处理 html 文本信息的函数，进行正则匹配等相关操作，以及获取所需信息如 Abstract，Pages，Publisher 等信息。例如以下代码就是获取摘要的部分：

```
re_Abstract = r'<div class="title3">Abstract</div>[\s\S]*?</p>'
abstract_list = re.findall(re_Abstract, this_html)
abstract_info = abstract(abstract_list)
```

在 Start_Scarp 函数中，指定起始文本页码，通过主函数传递的参数获取截至页码，实现对多网页的信息提取。

在主函数中，首先将检索后关键词带页码的网址作为 url_root，由于 Web of Science 网页的验证信息更新频繁，对于大量数据的读取不能一次性完成，因此在源码基础上修改代码。

将原来的一次性读取所有页码信息改为 while 循环指定次数，一次读取少批量信息多次读取，一旦出现连续多个网页没爬取就检查链接是否可靠，及时更迭。因此，每一次只爬取 100 个信息，存入一个表格，接着设置起始和截至页码，也以 100 为步长迭代。修改后代码如下：（根网址链接部分省去）

设置起始页码和截至页码以及存入的文件（也随 100 迭代），迭代为 100：

```
start_num = start_num + 100
doc_num = doc_num + 100
filename = "article_information/"+str(start_num)+"ML_material.paperInfo.csv"
```

爬取信息及设置表头

```
paper_Dic_Info = pagesToDic.Start_Scarp(root=url_root,nums_page=doc_num,
                 filename=filename, start_num=start_num)
info_dict = paper_Dic_Info

columns = ['title','Volume','Issue','Pages','whole__author_name',
           'simply_author_name','reprint author','DOI','reprint address',
           'Abstract','Keywords','Document Type',
           'Publisher','Research Domain','Published Date','impact_factor',
           'Keywords_plus','joural','pdf_link',"Download_SuccessOrDefeat"]
article_nums = len(info_dict)
```

填入表格

```
df = pd.DataFrame(index=range(1, article_nums), columns=columns)
for index in info_dict.keys():
    for column in info_dict[index].keys():
        if (type(info_dict[index][column]) == type([])):
            df.loc[index, column] = toString(info_dict[index][column], column)
        else:
            if column == 'reprint author':
                df.loc[index, column] = info_dict[index][column].replace(',', '')
            else:
                df.loc[index, column] = info_dict[index][column]
```

存盘

```
df.to_csv(filename)
```

爬取后获得多个表格文件，经过整合后，得到“Machine Learning”和“Material”的文献结果，表格作为附件提交。

1.2.3 未来展望

这学期偏向于对爬虫的初步了解和基本框架实现，注重应用于实际场景，对于正则匹配等提取 html 页面信息、beautifulsoup 解码等未进行深入了解，未来会完善细节的学习，逐步优化爬虫代码，用于今后数据集的获取。

3 体会和建议

本次创新实训就“自顶向下、逐步求精”算法设计和 Python 爬虫的理解和应用开展研究。

编程思想是编程的基础和灵魂，从代码结构就可以看出一位程序员的逻辑思路，逐步求精主要思想是贯彻先整体后局部，先暂时忽略细节完成框架，再完善的方法。本学期的“面向对象程序设计”就是建立在这样的基础上，对于特定的问题，先构建类的基本特征，再根据实际问题需求设计成员函数和数据成员；根据需求在类中定义其他需要的部件类，实现功能。由此可见，创新实训作为实践、拓展类的课程不是独立于专业课之外的，而是对于学科专业课也起到了指导作用，二者相辅相成，强化了我的编程能力和思想。通过对于“8 皇后问题”的实现，加强了对于程序设计结构的理解并在实际问题中得以实现。

优秀的代码是具有可读性、可重用性、可扩展性的，逐步求精就是在鼓励我们使程序分为多个部分，再细化。逐步求精的过程也是对于实际问题的抽象化为逻辑空间再具象化细节问题，在细化后，再用精确的定性思维寻找解决办法。细化的过程就是根据不同问题展开不同的延申和功能增加，也使得功能、函数模块尽量少的耦合，增加可读性，一个模块完成一个工作。

在 RoboMaster 机甲大师比赛中设计到对于敌方装甲板的识别与分类，本赛季将应用 yolov5 神经网络进行分类，对于初入神经网络、深度学习领域的我来说，这方面的知识十分需要。这学期的实训内容也具有一定的相关性，实训中导师就我们今后可能涉及到的数据挖掘领域首先指导我们应用 Python 爬虫获取论文数据，以便今后将知识符号化传递给计算机供其训练模型。在阅读 Dive-Into Deep Learning 的过程中，了解到比起凤毛麟角能研发出的高深算法，数据集获取的重要性同样重要，在信息繁杂的时代，筛选出需要的信息也成为了一项重要的技术。

上海交大在机甲大师比赛中就依靠其超强的数据挖掘实现目标预测，在赛场势不可挡；现实生活中，对于预测房价、大学录取率等热点问题，机器学习都具有相当大的施展空间。相信在未来，经过不断的学习，这项技术不仅可以帮助我在眼下的机器人领域更上一层楼，更会为我今后的职业生涯奠定基础。

小组内的同学都很优秀，和他们的交流中不仅可以学习到不同特色的思想，更是交到了有有趣灵魂的朋友，一个人可以很优秀，但一个团队总是强过一个人的，未来也希望可以和伙伴们一起学习，一起进步。

本科阶段想真正精通一项技术并不容易，我认为，更重要的是掌握一种可以终身收益的学习方法和逻辑思维，也就是一种自适应环境的能力。假如 5 年 10 年后人工智能的热潮过去，学到的技术过时，真正强大的人绝不会只是精通技术，而是会根据需求的不同不断学习所需的东西！这种能力才是本科、甚至大学阶段最值得培养的。我认为现在的创新实训以及和刘悦导师的沟通、对未来制定的计划都很契合上述这种能力的学习。学习，乃终身之务，绝非平时冶游，临近考试才匆匆拿出书本看两眼这么草率仓促，希望本科结束，我可以自豪地说，我做到了！

4 致谢

首先感谢的肯定是上海大学计算机工程与科学学院能开设这样一门颇有意义的课程，使学生的思维突破传统思维桎梏，培养学生的创新思维；其次是感谢刘悦导师悉心指导，作为我的全程导师，也时常关心我的学习生活、未来方向问题，提出了很多是我受益匪浅的指导意见，给我很大启发，感谢刘老师的付出和指点；同时，也感谢同组的优秀同学们，在和他们的交流中，解决了很多问题也学到了很多；最后，也感谢自己不断的努力，可能现在不是最优秀的，但每天比昨天进步就是成功！

5 参考文献

[1] Python爬虫系列1,载CSDN博客2021年2月24日,https://blog.csdn.net/qq_48322394/article/details/114014358?

[2] 八皇后问题，载百度百科，https://baike.baidu.com/item/八皇后问题/11053477

[3] 胡正国：程序设计方法学（第2版），国防工业出版社,第9页