深　圳 大 学 实 验 报 告

课程名称：　信 息 检 索

实验项目名称：词典、倒排记录表和容错式检索的实验

学院：　计 算 机 与 软 件 学 院

专业：　计 算 机 科 学 与 技 术

指导教师：　潘 微 科

报告人：　沈晨巧　学号：2019092121　班级：　1 9 计 科 0 4

实验时间：　2022年4月1日(周五)-2022年4月13日(周三)

实验报告提交时间：　２０２２年４月７日星期四

教务部制

实验目的与要求：

实验目的： 掌握跳表指针 (skip pointers) 技术、邻近搜索 (proximity search) 技术、编辑距离(editdistance) 计算方法、基于发音的矫正技术(phonetic correction) 等，了解PorterStemmer 开源程序的简单调用。

实验要求：

(1). 考虑利用如下带有跳表指针的倒排记录表

和两个中间结果表(如下所示，不存在跳表指针)分别进行合并操作。

3599699100101

2560120150

采用教材《Introduction to Information Retrieval》第37页 Figure 2.10 中所描述的基于跳表指针 (skip pointers) 的倒排记录表 (postings lists) 合并算法，请问：

a.跳表指针实际跳转的次数分别是多少(也就是说，指针 pl 的下一步将跳到 skip (pl))?

b.当两个表进行合并时，倒排记录之间的比较次数分别是多少?

c.如果不使用跳表指针，那么倒排记录之间的比较次数分别是多少?

请在报告中附上详细的文字说明。 (15分)

(2). 下面给出的是一个位置索引的一部分，格式为：

<position1,position2,…>;doc2:<positionl,position2,…>;…

angels:2:<36, 174, 252, 651>;4:<12, 22, 102, 432>;7:<17>;

fools:2:<1, 17, 74, 222>;4:<8, 78, 108, 458>;7:<3, 13, 23, 193>;

fear:2:<87, 704, 722, 901>;4:<13, 43, 113, 433>;7:<18, 328, 528>;

in:2:<3, 37, 76, 444, 851>;4:<10, 20, 110, 470, 500>;7:<5, 15, 25, 195>;

rush:2:<2, 66, 194, 321, 702>;4:<9, 69, 149, 429, 569>;7:<4, 14, 404>;

to:2:<47, 86, 234, 999>;4:<14, 24, 774, 944>;7:<199, 319, 599, 709>;

tread:2:<57, 94, 333>;4:<15, 35, 155>;7:<20, 320>;

where:2:<67, 124, 393, 1001>;4:<11, 41, 101, 421, 431>;7:<16, 36, 736>;

请问哪些文档和以下的查询匹配?其中引号内的每个表达式都是一个短语查询。

a.“angels fear”

b.“angels fear to tread”

c.“angels fear to tread”AND“fools rush in”

请在报告中附上详细的文字说明。 (10分)

(3). 阅读教材《Introduction to Information Retrieval》第37页 Figure 2.10 中所描述的基于跳表指针 (skip pointers) 的倒排记录表 (postings lists) 合并算法，并用Java 语言或其他常用语言实现该算法。要求在题(1)的例子上验证算法的正确性。

请在报告中附上代码截图(不要复制源代码，请用截图的方式)、运行结果截图和详细的文字说明。程序要有详细的注释。 (10分)

(4). 阅读教材《Introduction to Information Retrieval》第42页 Figure 2.12 中所描述的邻近搜索(proximity search) 中的两个倒排记录表(postings lists)的合并算法，并用 Java 语言或其他常用语言实现该算法，并按要求做适当改进。要求使用附件“HW2.txt” 中的

60个文档(每行表示一个document,按空格切词，文档中的单词全部转换为小写)建立positional index,两个词项之间的间距(注：相邻的两个词项的间距为1)的形式包括以下三种情形(x是一个正整数)：

“-x”、“+x”和“x”，

其中，“-x”表示第一个词项在第二个词项的左侧且间隔在x之内，“+x”表示第一个词项在第二个词项的右侧且间隔在x之内，“x”表示第一个词项与第二个词项的间隔(左侧和右侧均可)在x之内。要求在以下例子上验证算法的正确性：(ranking,filtering,-4)

(ranking,filtering,-5),(ranking,filtering,-6),(ranking,filtering,-7),(heterogeneous,learning,+2),(recommendation,bias,2)。

请在报告中附上代码截图(不要复制源代码，请用截图的方式)、运行结果截图和详细的文字说明。程序要有详细的注释。(30分)

(5).阅读教材《Introduction to Information Retrieval》第59页Figure 3.5中所描述的基于动态规划(dynamic programming)来计算两个字符串的编辑距离(edit distance)的算法，并用Java语言或其他常用语言实现该算法。要求计算以下15组单词的编辑距离：

(business,buisness)

(committee,commitee)

(conscious,concious)

(definitely,definately)

(fluorescent,florescent)

(forty,fourty)

(government,goverment)

(idiosyncrasy,idiosyncracy)

(immediately,immediatly)

(millennium,millenium)

(noticeable,noticable)

(tendency,tendancy)

(truly,truely)

(weird,wierd)

(privilege,privledge)

请在报告中附上代码截图(不要复制源代码，请用截图的方式)、运行结果截图和详细的文字说明。程序要有详细的注释。(15分)

报告写作。要求：主要思路有明确的说明，重点代码有详细的注释，行文逻辑清晰、可读性强，报告整体写作较为专业。(20分)

说明：

(1)本次实验课作业满分为100分。

(2)本次实验课作业截至时间2022年4月13日(周三)22:00。

(3)报告正文：请在指定位置填写，本次实验需要单独提交源程序文件(源程序单独打包在Blackboard中上传，不要包含外部导入的包)。

(4)个人信息：WORD文件名中的“姓名”、“学号”,请改为你的姓名和学号；实验报告的首页，请准确填写“学院”、“专业”、“报告人”、“学号”、“班级”、“实验报告提交时间”等信息。

(5)提交方式：截至时间前，请在Blackboard平台中提交。

(6)发现抄袭(包括复制&粘贴整句话、整张图),**抄袭者和被抄袭者的成绩记零分**。

(7)延迟提交，不得分；如有特殊情况，请于截至日期之后的48小时内发邮件到panweike@szu.edu.cn,并在邮件中注明课程名称、作业名称、姓名、学号等信息，以及特殊情况的说明，我收到后会及时回复。

(8)期末考试阶段补交无效。

(1). 考虑利用如下带有跳表指针的倒排记录表

和两个中间结果表(如下所示，不存在跳表指针)分别进行合并操作。

3599699100101

2560120150

采用教材《Introduction to Information Retrieval》第37页 Figure 2.10 中所描述的基于跳表指针 (skip pointers) 的倒排记录表 (postings lists) 合并算法，请问：

a.跳表指针实际跳转的次数分别是多少(也就是说，指针 pl 的下一步将跳到 skip (pl))?

b.当两个表进行合并时，倒排记录之间的比较次数分别是多少?

c.如果不使用跳表指针，那么倒排记录之间的比较次数分别是多少?

请在报告中附上详细的文字说明。 (15分)

++

a.跳表指针实际跳转的次数分别是多少(也就是说，指针 p1的下一步将跳到 skip(p1))?

跳表指针的跳转算法为：如果当前p1 节点拥有跳表指针，当 pl. val 小于p2. val 时不断向后跳转。 (具体比较过程可以见第2题，或者看代码实现)

1、和3599699100101进行合并操作：

答： **2 次，**<24, 75>,<75, 92>

比较到24与96时， 24拥有 skip指针且75<96, 因此第1次跳转；

75拥有skip 指针且92<96, 因此第2次跳转；

92拥有 skip 指针但是115>96, 结束跳转。

2、和2560120150进行合并操作：

答：**3 次，**<3, 24>,<75, 92>,<92, 115>

比较到3与25时， 3拥有skip 指针且24<25, 因此第1次跳转；

24拥有 skip 指针且75>25, 结束跳转。

比较到75与120时， 75拥有 skip指针且92<120, 因此第2次跳转；

92拥有skip 指针且115<120, 因此第2次跳转；

115无 skip 指针，结束跳转。

++

b.当两个表进行合并时，倒排记录之间的比较次数分别是多少?

当pl.val =p2.val 时，加入答案列表，并且plp2 同时后移；当 pl.val<p2.val 时，如果 pl 无 skip 指针，则向后移，否则根据第1题的跳表指针算法向后移动 pl 指针；反之同理。(具体过程可以看代码实现)

1、和3599699100101进行合并操作：

答： **13次，**比较顺序如下： <pl.val,p2.val>

[<3, 3>, '<5, 5>', '<9, 9>, '<15, 96>, '<24, 96>', '<75, 96>, '<92, 96>', '<115, 96>', '<96, 96>, '<97, 99>', <100, 99>, <100, 100>', <115, 101>]

2、和2560120150 进行合并操作：

答： **10 次，**比较顺序如下：<pl.val,p2.val>

[<3, 25>', '<24, 25>', '<75, 25>, <39, 25>', '<39, 60>, '<60, 60>', '<68, 120>', '<75, 120>', <92, 120>', '<115, 120>]

++

c.如果不使用跳表指针，那么倒排记录之间的比较次数分别是多少?

1、和3599699100101进行合并操作：

答： **18 次，**比较顺序如下： <pl.val,p2.val>

[<3, 3>, '<5, 5>, '<9, 9>, <15, 96>', <24, 96>', '<39, 96>, '<60, 96>', '<68, 96>', '<75, 96>', '<81, 96>', '<84, 96>', <89, 96>', '<92, 96>', <96, 96>', '<97, 99>', '<100, 99>', <100, 100>', '<115, 101>]

2、和2560120150进行合并操作：

答 **：18 次，**比较顺序如下： <p1.val,p2.val>

[<3, 25>, '<5, 25>, <9, 25>', '<15, 25>, <24, 25>', '<39, 25>, '<39, 60>', '<60, 60>', '<68, 120>', '<75, 120>', <81, 120>, <84, 120>', '<89, 120>', '<92, 120>, '<96, 120>', '<97, 120>', <100, 120>', '<115, 120>]

(2). 下面给出的是一个位置索引的一部分，格式为：

<position1,position2,…>;doc2:<position1,position2,…>;…

angels:2:<36, 174, 252, 651>;4:<12, 22, 102, 432>;7:<17>;

fools:2:<1, 17, 74, 222>;4:<8, 78, 108, 458>;7:<3, 13, 23, 193>;

fear:2:<87, 704, 722, 901>;4:<13, 43, 113, 433>;7:<18, 328, 528>;

in:2:<3, 37, 76, 444, 851>;4:<10, 20, 110, 470, 500>;7:<5, 15, 25, 195>;

rush:2:<2, 66, 194, 321, 702>;4:<9, 69, 149, 429, 569>;7:<4, 14, 404>;

to:2:<47, 86, 234, 999>;4:<14, 24, 774, 944>;7:<199, 319, 599, 709>;

tread:2:<57, 94, 333>;4:<15, 35, 155>;7:<20, 320>;

where:2:<67, 124, 393, 1001>;4:<11, 41, 101, 421, 431>;7:<16, 36, 736>;

请问哪些文档和以下的查询匹配?其中引号内的每个表达式都是一个短语查询。

a.“angels fear”

b.“angels fear to tread”

c.“angels fear to tread”AND“fools rush in”

请在报告中附上详细的文字说明。 (10分)

++

a.“angels fear”

1. 符合查询的文档： **文档 4** 中 ，angles-12 fear-13 或 angles-432 fear-433

文档 7中 ，angles-17 fear-18

++

b.“angels fear to tread”

1. 符合查询的文档： **文档 4** 中 ，angles-12 fear-13 to -14 tread -15

++

c.“angels fear to tread”AND“fools rush in”

1.“fools rush in”符合的文档：文档2中，fools-1rush-2 in-3

文档4中， fools-8 rush-9 in-10

文档7中，fools-3 rush-4in-5或fools-13 rush-14 in-15
2．与第二组结果取AND,因此最终结果为**文 档 4**

(3).阅读教材《Introduction to Information Retrieval》第37页Figure 2.10中所描述的基于跳表指针(skip pointers)的倒排记录表(postings lists)合并算法，并用Java语言或其他常用语言实现该算法。要求在题(1)的例子上验证算法的正确性

请在报告中附上代码截图(不要复制源代码，请用截图的方式)、运行结果截图和详细的文字说明。程序要有详细的注释。(10分)

++

代码截图和详细的文字说明：

首先定义一个数据结构

链表节点Node

```
class  Node:
  def_init_(selt,val):
    selfval  =val
    self.next  =None
    self.skip=None
```

值得注意的是，添加了一个skip指针。如果该节点可以跳转，则skip指针指向下一个节点，否则为None

定义一个含跳表指针的链表：

其中包括了三个方法：

1.根据list添加节点

2.添加单节点

3.添加跳表指针

前两个方法为常见的链表算法。

```
class  linkedListWithSkips:
 def_init_(self):
   selt.head  =None
   self.length=0

 def  addNodes(self,list):
   #将list中的元素添加到链表中
   for  iin  list:
     self.addNode(()

  def  addNode(self,val):
   #添加一个节点
   if  self.head  ==None:
     self.head  =Node(val)
      self.length  +=1

   else:
     CUrr =selfheac
     while  curr.next  !=None:
      curr  =curr.next
     curr.next=Node(val)
      self.length  +=1
```

第三个方法实现如下：

首先根据链表长度计算出skip 指针的跳转长度为 interval =sqrt(length)

然后从第一个节点开始，每隔interval个节点指向下一个跳转节点。

```
def addskips(sell):
#interval 为间阐，每个间隔之后添加一个skip指针
#interval的大小根据长度开根号决定
interval =math.fioor(math.sgrt(self.length))
#每隔interval个节点，添加一个skip指针
cur =self.head
curr =self.head
while curr.next !=None:
    foriin range(interval):
        if curr.next!=None:
            curr =curr.next
        else;
            break
    #如果cur后存在interval个节点，则添加skip 指针
    ifi==interval-1:
        cur.skip=cur
        cur =curr
```

合并算法：（含跳表指针）

```
def intersectWithSkips(pi,p2):
ans =
while p¹!=None and p2!=None:
    #如果val相同，则加入答案，指针后移
    if p¹val ==p2.vat
        ans.append(pival)
        P1=pl.next
        P2=p2.next
    #val较小的指针，如果存在skp 指针，则不断向后跳直到val较大为止
    elf plval<p2.vat:
        #存在skip指针，则不断向后比较
        if pl.skip and p¹.skipval<=p2vat
            while pi.skip and p¹.skipval<=p2.val:
                p¹=pl.skip
        else:
            p1=p¹.next
    #同上
    elf plval>p2vak
        if p2.skip and p2.skipval <=pival:
            while p2.skip and p2.skipval <=p¹.val:
                P2=p2.skip
        else:
            p2=p2.next
print('ans:,ans)
```

（不含跳表指针）

```
def intersectWithoutSkips(p¹,p2):
    ans =
    while p¹!=None and p2 !=None:
        # 如果val相同，则加入答案，指针后移
        if plval ==p2val:
        ans.append(p¹.val)
        p1=p¹.next
        p2=p2.next
        #val 较小的指针，如果存在skip指针，则不断向后跳直到val较大为止
        elif plval<p2.val:
        p1=p¹.next
        elif p¹val>p2.val:
        p2=p2.next
    print(ans:',ans)
```

统计次数：

```
ef intersectWithSkips_stat(pl,p²):
ns=1
kipcount,comparecount=0,0
kip_aray,compare_aray=[.1
while p¹!=None and p2!=None:
#如果vdl相同，则加入答案，指针后移
    plval==p2.val
omparecount+=1
ompare_arayappend(<'+str(plval)+!'+str(p2val)+'>)
ns.append(pival)
¹1=pl.next
²2=p2.next
#al 较小的指针，如果存在水p 指针，则不断向后跳直到va较大为止
lf plval<p2vat
comparecont4=1
ompare_arayapPend(<'+str(plval)+i'+str(p2val)+'>)
while pl.skip
omparecount +=1
ompare_arrayappend(<'+sr(plskipval)+!'+str(p²val)+>)
p¹.skipval<=p2.vak
kipcount+=1
kip_arrayappend('<'+str(plval)+''+str(pl.skipval +>)
¹1=p¹.skp
else;
break
l=p1.next
lf pival>p2vat
omparecount +=1
ompare_arayappend(<'+str(pival)+!'+str(P2val)+'>)
while p2.skip:
omparecount +s1
ompare_arrayappend(<+str(plval)+f'+str(p2skipval)+'>)
2.skipval<=pivak
kipcount +=1
kip_aray.aPPend(<'+strp²val)+!'+st(p2.skipval)+>)
²2=p2.skip
else:
break
²2=P2.next

rint(comparecount',comparecount)
rintf compare_aray:,compare_aray)
rint(skipcountt,skpcount)
int(skip_aray:;,skip_aray)
```

```
def      intersectWithoutSkips_stat(pl,p2):
  ans=1
  comparecount,compare_aray=0,D
  while   p'!=None   and   p2!=None;
    comparecount   +s7
    compare_arrayappend(<'+str(plval)+;'+str(p2val)+'>)
    # 如 果va 相同，则加入答案，指针后移
    if plval ==p2.vat
      ans.append(pival)
      p1=pl.next
      p2=p2.next
    #val 较小的指针，如果存在skip指针，则不断向后跳直到val较大为止
    eltpival<p2vak
      p1=pl.next
    ellfpival>p2.vat
      p2=p2.next
  print(comparecount',comparecount)
  print('compare_array!,compare_aray)
```

++

运行结果截图和详细的文字说明(1)：

带有跳表指针的倒排记录表

和3599699100101 的合并操作

定义两个链表，并为list1 加 入skip 指 针 ，list2 不加 skip 指 针

```
isti =linkedListWithSkips0
list¹.addNodes([3, 5, 9, 15, 24, 39, 60, 68, 76,
        81, 84, 89, 92, 96, 97, 100, 115])
ist¹.adaSkips()
ist2 =linkedListWithSkips)
ist2.addNodes([3, 5, 9, 96, 99, 100, 101])
```

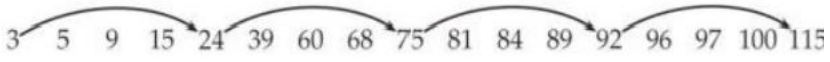
调用函数，计算合并结果并输出统计结果

```
集
ans  [3, 5. 9, 96, 100
comparecount  13
 nPre-Y [<3,3>,<5,5>,'<9,9>|<15,96>;<24,96>,'95,96>,<92,96><115,96>'<96,96>,<97,99><100,99>;<100,100>;<115,10>
skipcount:2
akp  aray:[<24,769](<7592>]
```

++

运行结果截图和详细的文字说明(2)：

带有跳表指针的倒排记录表



和2560120150 的合并操作

定义 list3 链表，不加入 skip 指针

```
ist3 =linkedListWithSkips ()
list3.addNodes([25,60,120,150])
```