

# 1 计算题

## 1.1

1.1 请推荐如下查询的处理次序。

(tangerine OR trees) AND (marmalade OR skies) AND (kaleidoscope OR eyes)

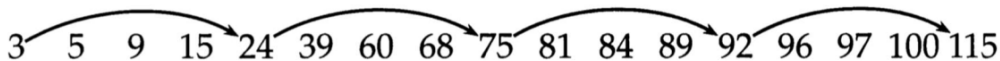
其中，每个词项对应的倒排记录表的长度分别如下：

词项	倒排记录表长度
eyes	213 312
kaleidoscope	87 009
marmalade	107 913
skies	271 658
tangerine	46 653
trees	316 812

- 已由题目信息得到每个词项的文档频率
- 从左至右估计每个OR操作后的结果大小（按最坏情况）
  - tangerine OR trees :  $46653+316812=363465$
  - marmalade OR skies :  $107913+271658=379571$
  - kaleidoscope OR eyes:  $87009+213312=300321$
- 按OR结果大小，从小到大顺序执行AND
  - 先执行 (kaleidoscope OR eyes) AND (tangerine OR trees)
  - 所得结果b再执行 b AND (marmalade OR skies)

## 1.2

1.2 考虑利用如下带有跳表指针的倒排记录表



和一个中间结果表（如下所示，不存在跳表指针）进行合并操作。


3 5 89 95 97 99 100 101

采用基于跳表指针的倒排记录表合并算法，请问：

- 1) 跳表指针实际发生跳转的次数是多少？
- 2) 当两个表进行合并时，倒排记录之间的比较次数是多少？
- 3) 如果不使用跳表指针，那么倒排记录之间的比较次数是多少？

采用基于跳表指针的合并算法流程（记两表的指针为p1,p2，图示为算法的伪代码）

INTERSECTWITHSKIPS(p1,p2)

```
1  answer ← 
2  while p1 ≠ NIL and p2 ≠ NIL
3  do if docID(p1) = docID(p2)
4      then ADD(answer, docID(p1))
5          p1 ← next(p1)
6          p2 ← next(p2)
7  else if docID(p1) < docID(p2)
8      then if hasSkip(p1) and (docID(skip(p1)) ≤ docID(p2))
9          then while hasSkip(p1) and (docID(skip(p1)) ≤ docID(p2))
10             do p1 ← skip(p1)
11             else p1 ← next(p1)
12      else if hasSkip(p2) and (docID(skip(p2)) ≤ docID(p1))
13          then while hasSkip(p2) and (docID(skip(p2)) ≤ docID(p1))
14             do p2 ← skip(p2)
15      else p2 ← next(p2)
16  return answer
```

<https://blog.csdn.net/dongjishuo>

1. 3与3比较，加入answer集，p1++,p2++
2. 5与5比较，加入answer集，p1++,p2++
3. 9与89比较，p1++
4. 15与89比较，p1++
5. 24与89比较，并发现24处跳表指针75小于89，所以p1指向75；再发现75处跳表指针92大于89，所以不跳
6. 75与89比较，并发现75处跳表指针92>89，所以p1++
7. 81与89比较，p1++
8. 84与89比较，p1++
9. 89与89比较，加入answer集，p1++,p2++
10. 92与95比较，发现92处跳表指针115>95，所以p1++
11. 96与95比较，p2++
12. 96与97比较，p1++
13. 97与97比较，加入answer集，p1++,p2++
14. 100与99比较，p2++
15. 100与100比较，加入answer集，p1++, p2++
16. 101与115比较，p2++，结束

不用跳表指针的合并算法流程（算法伪代码见ppt）

1. 3与3比较，加入answer集，p1++,p2++
2. 5与5比较，加入answer集，p1++,p2++
3. 9与89比较，p1++
4. 15与89比较，p1++
5. 24与89比较，p1++
6. 39与89比较，p1++
7. 60与89比较，p1++
8. 68与89比较，p1++
9. 75与89比较，p1++
10. 81与89比较，p1++
11. 84与89比较，p1++
12. 89与89比较，加入answer集，p1++,p2++
13. 92与95比较，p1++
14. 96与95比较，p2++
15. 96与97比较，p1++

16. 97与97比较，加入answer集，p1++,p2++
17. 100与99比较，p2++
18. 100与100比较，加入answer集，p1++, p2++
19. 101与115比较，p2++，结束

问题回答

1. 跳表指针实际跳转次数：1次（24->75）
2. 倒排记录间比较次数：16+4=20次(4次是与跳表指针比较)
3. 不用跳表指针，比较次数：19次（跳表指针在时恰跳过了三个元素）

## 1.3

1.3 写出倒排记录表（777, 17743, 294068, 31251336）的可变字节编码。在可能的情况下对间距而不是文档 ID 编码。写出 8 位块的二进制码。

文档ID	777	17743	294068	31251336
间距		16966	276325	30957268
VB编码	00000110	00000001	00010000	00001110
	10001001	00000100	01101110	01100001
		11000110	11100101	00111101
				11010100

## 2 简答题

2.1 基于机械分词的常见方法中对于“最大匹配”的依赖，可能导致什么隐患？如何利用 N-最短路径缓解这一隐患？如何选择一个恰当的 N 值

- 隐患：
  - 维护高质量词典需要极大的开支
  - 永远难以应对新生词汇
  - 词汇频率/重要性往往对结果不产生影响
  - 对于有歧义的句子，有多种切分方式的句子，“最大匹配”可能得不到正确结果
- 利用N-最短路径：
  - 保留N条最短路径，以提供更多分词方案，能更好的消除歧义
  - 而且可对边施加权重，提供基于统计的分词方法，结合词汇频率及重要性来作切分
- 选择恰当的N值
  - 统计大量句子分词时最终选取是第n短路径，取这些n值的平均后可得到一个N值

- 或是通过机器学习、深度学习方法，构造包含N-最短路径算法的模型，利用人工标注了句子正确分词方式的训练集进行训练，确定N-最短路径算法中的参数N
- 或是考虑设置较大的N值来训练词的边权重：对找到的N条最短路径，按照正确切分方式人工对边权重做新标注。

在实际使用时即可设置较小的N值来节省存储空间，减少人力复核消耗等，同时因为前面做过一轮边权重标注，所以又能保证一定的准确度。

## 2.2 如何结合查询词项的分布细节，设计相对合理的跳表指针步长

- 考虑求词项A AND B
- 对一段文档ID的取值区间
  - 若A、B对应的倒排表中的文档ID在该区间都比较密集，则均设置步长相对小的跳表指针
  - 若A、B对应的倒排表中的文档ID在该区间都比较稀疏，则均设置步长相对大的跳表指针或不用在该区间设置跳表指针
  - 若A对应的倒排表中文档ID在该区间更密集，B在该区间的文档ID间隔都较大，则考虑只在A中设置较大步长的跳表指针

## 2.3 在信息检索系统中，如何同时使用位置索引和停用词表？潜在问题有哪些，如何解决？

- 如何同时使用：
 

一般方法是先对词典用停用词表过滤一遍，再在倒排表构建里使用位置索引，这样可以节省大量空间，因为停用词多为高频词，在大量文档中出现，提前过滤掉停用词可以让倒排表所需存储空间大大减小。

对于已经利用位置索引建立好的倒排表，若还含有大量停用词，可以考虑用停用词表来对其过滤。
- 潜在问题：
  - 某些停用词在特定场景下有意义
  - 某些停用词的组合有意义
  - 位置索引对存储空间需求较大，因为还要存位置信息
- 解决方案
  - 进一步可以考虑，先用位置索引的方法构建倒排表，然后利用有实际意义的停用词组合构成的表来对倒排表作过滤
 

比如：对于停用词的组合：to be or not to be, 考虑对to 和be的倒排表作检查，利用其位置信息，若to 和 be在同一文档中的位置符合这样一种关系，则考虑不作去除。没有这样特殊关系的停用词则去除
  - 当然，上述方式在初始构建倒排表时开销较大。所以也可以借鉴多元词索引的方法，把有意义的停用词组作为一个词汇加入词汇表，在构建倒排表的过程中，就可以把无实际意义的停用词去掉。
  - 在存储空间方面，考虑利用索引压缩技术，分别从词汇表和倒排表入手作压缩。具体到位置索引的使用上，可以考虑把存储位置信息也改成存储位置信息间的距离，并利用可变长度编码。

