Chapter I

Exercise 1.1

Draw the inverted index that would be built for the following document collection:
Doc 1 new home sales top forecasts
Doc 2 home sales rise in july
Doc 3 increase in home sales in july
Doc 4 july new home sales rise

forecast	Doc 1			
home	Doc 1	Doc 2	Doc 3	Doc 4
increase	Doc 3			
july	Doc 2	Doc 3	Doc 4	
new	Doc 1	Doc 4		
rise	Doc 2	Doc 4		
sale	Doc 1	Doc 2	Doc 3	Doc 4
top	Doc 1			

Exercise 1.2

Consider these documents:
Doc 1 breakthrough drug for schizophrenia
Doc 2 new schizophrenia drug
Doc 3 new approach for treatment of schizophrenia
Doc 4 new hopes for schizophrenia patients
Question:

a) Draw the term-document incidence matrix for this document collection.
b) Draw the inverted index representation for this collection, as in Figure 1.3 (page 7).

Answer:

a) Term document incidence

Term/Document	Doc 1	Doc 2	Doc 3	Doc 4
approach	0	0	**1**	0
breakthrough	**1**	0	0	0
drug	**1**	**1**	0	0
hope	0	0	0	**1**
new	0	**1**	**1**	**1**
patient	0	0	0	**1**
schizophrenia	**1**	**1**	**1**	**1**

treatment	0	0	**1**	0

b) Inverted index

approach	3			
breakthrough	1			
drug	1	2		
hope	4			
new	2	3	4	
patient	4			
schizophrenia	1	2	3	4
treatment	3			

Exercise 1.3

For the document collection shown in Exercise 1.2, what are the returned results for these queries:

a) schizophrenia AND drug
b) for AND NOT (drug OR approach)

answer

a) Doc 1, Doc 2
b) Doc 4

Exercise 1.4

For the query below, can we still run through the intersection in time O(x+y), where x and y are the lengths of the postings lists for Brutus and Caesar? If not, what can we achieve?

a) Brutus AND NOT Caesar
b) Brutus OR NOT Caesar

answer

a) Yes, we can still run O(x+y) as long as any postings of Caesar which matched with postings of Brutus is not intersected. Or in other, the length of intersected posting list is less than or equal to length of postings of Brutus.
b) No, we can not achieve O(x+y) in x OR NOT y operation, since the length of NOT y is unknown and same as length of intersect posting list. We can optimize this query into O[x+y'], where y' is length of postings which not existed in x and y.

Exercise 1.5

Extend the postings merge algorithm to arbitrary Boolean query formulas. What is its time complexity? For instance, consider:

a) (Brutus OR Caesar) AND NOT (Antony OR Cleopatra)

Can we always merge in linear time? Linear in what? Can we do better than this?

Answer:

a) Yes, we can always merge in linear time as input size going to be bigger since we need to find any postings which matched with any member in merged result of (Brutus OR Caesar), but not match in any posting in both Antony AND Cleopatra. We can do better, by only take scan on postings which not included in merge result of (Brutus OR Caesar) and hope the merge result of (Brutus OR Caesar) always long.

Exercise 1.6

We can use distributive laws for AND and OR to rewrite queries.

a) Show how to rewrite the query in Exercise 1.5 into disjunctive normal form using the distributive laws.
b) Would the resulting query be more or less efficiently evaluated that the original form of this query?
c) Is this result true in general or does it depend on the word and the contents of the document collection?

Answer

a) $(Brutus \vee Caesar) \wedge \neg(Antony \vee Cleopatra) \equiv (Brutus \vee Caesar) \wedge (\neg Antony \vee \neg Cleopatra)$
b) Would be less efficiently because we need to produce each negation first before operation OR.
c) It is true for general.

Exercise 1.7

Recommend a query processing order for

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

given the following postings list sizes:

Term	Postings Size
eyes	213312
kaleidoscope	87009
marmalade	107913
skies	271658
tangerine	46653
trees	316812

Answer:

N(tangerine) + N(trees) = 363465
N(marmalade) + N(skies) = 379571
N(kaleidoscope) + N(eyes) = 1083321

Initial query is already omptimum.
(tangerine OR trees) AND (marmalade OR skies) AND (kaleidoscope OR eyes)

Exercise 1.8

If the query is:
b) friends AND romans AND (NOT countrymen)

How could we use the frequency of countrymen in evaluating the best evaluation order? In particular, propose a way of handling negation in determining the order of query processing.

Answer:
The merge result from this query is any intersected term from friends AND romans that **not matched** with any terms found in countrymen. The result length should be smaller than or equal to the length of friend AND romans.

Exercise 1.9

For a conjunctive query, is processing postings lists in order of size guaranteed to be optimal? Explain why it is, or give an example where it isn't.

Answer:
Processing conjunctive query in order of size is not guarantee to be optimal because the distributive law proof that addition in any direction has always produce exactly same result and the representation of linear growth tell that any addition function of size is always exactly product total of length of its element. For example, (a + b) + c is equivalent with a + (b + c).

Heading 1.10

Write out a postings merge algorithm, in the style of Figure 1.6 (page 11), for an x OR y query.

Answer:

```
INVERTED_INDEX *px = x
INVERTED_INDEX *py = y
INVERTED_INDEX *z = allocate()

while (*px).next != null || (*py).next != null:
  if (*px).docId != null && (*py).docId != null:
    if (*px).docId == (*py).docId:
      (*z).docId = (*px).docId
      px = (*px).next
      py = (*py).next
    else:
      if px.docId < py.doId:
        (*z).docId = (*px).docId
        px = (*px).next
      else:
        (*z).docId = (*py).docId
        py = (*py).next
  else:
    if px != null:
      (*z).docId = (*px).docId
    else:
      (*z).docId = (*py).docId

  (*z).next = alocate()
```

```
  z = (*z).next

&z = null
```

Exercise 1.11

How should the boolean query x AND NOT y be handled? Why is naive evaluation of this query normally very expensive? Write out a postings merge algorithm that evaluates this query efficiently.

Answer:
Find any terms of x that not matched with terms of y. The algorithm may be like this:

```
INVERTED_INDEX *px = x
INVERTED_INDEX *py = y
INVERTED_INDEX *z = allocate()

while px != null:
  if (*px).docId < (*py).docId:
    (*z).docId = (*px).docId
    px = (*px).next
  else:
    py = (*py).next

  (*z).next = allocate()
  z = (*z).next

&z = null
```

Exercise 1.12

Write a query using Westlaw syntax which would find any of the words professor, teacher, or lecturer in the same sentence as form of the verb explain.

Answer:
(professor! Teacher! Lecturer!) /s

Exercise 1.13

Try using boolean search features on a couple of major web search engines. For instance, choose a word, such as burglar, and submit the queries (I) burglar, (ii) burglar AND burglar, and (iii) burglar OR burglar. Look at estimated number of results and top hits. Do they make sense in terms of Boolean logic? Often they have not for major search engines. Can you make sense of what is going on? What about if you try different words? For example, query for (I) knight. (ii) conquer, and then (iii) knight OR conquer. What bound should the number of results from the first two queries place on the third query? Is this bound observed?

Exercise 2.1

Are the following statetemnts true or false?

a) In a Boolean retrieval system, stemming never lowers precision. (False)
b) In a Boolean retrieval system, stemming never lowers recall. (True)

c) Stemming increases the size of the vocabulary. (False)
d) Stemming should be invoked at indexing time but not while processing a query. (False)

Exercise 2.2

Suggest what normalized form should be used for these words (including the word itself as a possibility):

a) 'Cos, cos, because
b) Shi'ite, shiite
c) cont' d, count down, counting down
d) Hawai'i, hawai
e) O'Rourke, o rourke, rourke

Exercise 2.3

The following pairs of words are stemmed to the same form by Porter stemmer. Which pairs would you argue shouldn't be conflated. Give your reasoning:

a) Abandon / abandoment. This associative is good because term abandoment is clearly adjective form of verb abandon.
b) Absorbency / absorbent. This associative looks fine.
c) Marketing / markets. This associative looks not relevant because markets refer to place, while marketing refer to job function.
d) University / universe. This associative clearly not relevant, because term universe has general meaning while term university refers to place.
e) Volume / volumes. This associative looks really good.

Exercise 2.4

For the Porter stemmer rule group shown in (2.1):

a) What is the purpose of including an identity rule such as SS → SS?
   To identify such term belong to adjective.
b) Applying just this rule group, what will the following words be stemmed to?

circus	*canaries*	*boss*
circus	canari	boss

c) What rule should be added to correctly stem *pony*?
   IESS → Y
d) The stemming for *ponies* and *pony* might seem strange. Does it have a deleterious effect on retrieval? Why or why not?
   No, it does not have high deleterious effect on retrieval result because *poni* as result of stemming of *ponies* and *pony* have been stored at index at first will always matched with *poni* from post processed term of query.