**Sheet 1**

**Exercise 1**

For the queries 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

**Exercise 2**

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?

**Exercise 3**

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

**a.** Show how to rewrite the query in Exercise2 into disjunctive normal form using the distributive laws.

**b.** Would the resulting query be more or less efﬁciently evaluated than the original form of this query?

**c.** Is this result true in general or does it depend on the words and the contents of the document collection?

**Exercise 4**

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

**Exercise 5**

If the query is:

**a.** friends AND romans AND (NOT countrymen)

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

**Exercise 6**

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.

**Exercise 7**

Write out a postings merge algorithm ,for an x OR y query.

**Exercise 8**

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.