# TD 03

## Exercise 01

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?
– Brutus AND NOT Caesar
– Brutus OR NOT Caesar

Extend the postings merge algorithm to arbitrary Boolean query formulas. What is its time complexity? For instance, consider:
- (Brutus OR Caesar) AND NOT (Antony OR Cleopatra)

**SOLUTION.** a. Time is O(x+y). Instead of collecting documents that occur in both postings lists, collect those that occur in the first one and not in the second. b. Time is O(N) (where N is the total number of documents in the collection) assuming we need to return a complete list of all documents satisfying the query. This is because the length of the results list is only bounded by N, not by the length of the postings lists.

**SOLUTION.** We can always intersect in $O(qN)$ where q is the number of query terms and N the number of documents, so the intersection time is linear in the number of documents and query terms. Since the tightest bound for the size of the results list is $N$, the number of documents, you cannot do better than $O(N)$.

## Exercise 02

Write a merge algorithm in the style of x OR y.

**SOLUTION.** UNION(x, y)

```
1answer<- ( )
2while x!=NIL and y!=NIL
3do if docID(x)=docID(y)
4 then ADD(answer,docID(x))
5 x<- next(x)
6 y<-next(y)
7 else if docID(x)<docID(y)
8 then ADD(answer,docID(x))
9 x<- next(x)
10 else ADD(answer,docID(y))
11 y<-next(y)
12 return(answer)
```

## Exercise 03

How should the Boolean query x AND NOT y be handled? Describe a naïve algorithm for this and then write an efficient solution

**SOLUTION.**

A naive evaluation of the query x AND (NOT y) would be to calculate (NOT y) first as a new postings list, which takes O(N) time, and the merge it with x. Therefore, the overall complexity will be O(N).

An efficient postings merge algorithm to evaluate x AND (NOT y) is:

```
MERGE(x, y, AND NOT)
1answer<- ( )
2while x!=NIL and y!=NIL
3do if docID(x)=docID(y)
4 then x<- next(x)
5 y<-next(y)
6 else if docID(x)<docID(y)
7 then ADD(answer,docID(x))
8 x<- next(x)
9 else y<-next(y)
10 return(answer)
```

Exercise 04

We have two-word query. For one term the postings list consists of the following 16 entries:
[4,6,10,12,14,16,18,20,22,32,47,81,120,122,157,180]
And for the other it is the one entry posting list: [47]
Work out how many comparisons would be done to intersect the two posting lists?

**SOLUTION.**

Applying MERGE on the standard postings list, comparisons will be made unless either of the postings list end i.e. till we reach 47 in the upper postings list, after which the lower list ends and no more processing needs to be done.
Number of comparisons = 11

b. Using skip pointers of length 4 for the longer list and of length 1 for the shorter list, the following comparisons will be made: 1. 4 & 47 2. 14 & 47 3. 22 & 47 4. 120 & 47 5. 81 & 47 6. 47 & 47 Number of comparisons = 6

Exercise 05

**SOLUTION.** Answer (a): All three documents (2, 4, and 7) satisfy the query. Answer (b): Only document 4.