Query Processing with an Inverted Index

Contents

- how inverted index is an efficient data structure for doing query operations in an IR system.
- how you can perform a common kind of query, an AND query for two terms.

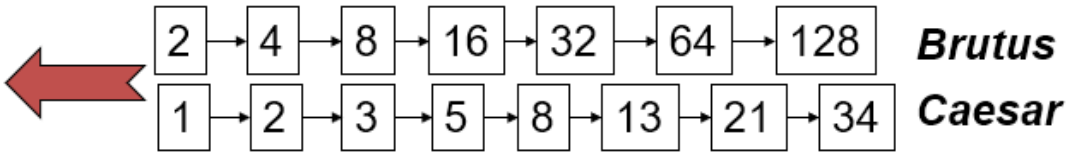
  → details of query processing

Query Processing : AND

- consider processing the query:

  Brutus and **Caesar**

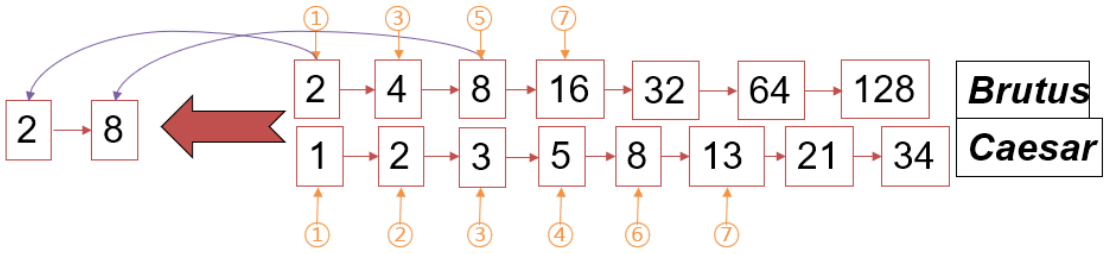
  - locate **Brutus** in the Dictionary :
    - Retrieve its postings
  - locate **Caesar** in the Dictionary :
    - Retrieve its postings
  - "Merge" the two postings (intersect the document sets)



the Merge

- walk through the two postings simultaneously, in time linear in the total number of postings entries
  - 수행 단계

  1. start with a pointer which points at the head of both lists
  2. ask are these two pointers pointing at the same, an equal doc ID
  3. (3을 수행 후, 2로 돌아감)
     1. if the answer is no, then advance the pointer that has the smaller doc ID.
     2. if the answer if yes, put the doc ID into our result list
  4. if one of postings lists is exhausted, then stop & return result list (documents)



  - 알고리즘

Intersecting two postings lists (a “merge” algorithm)

```
INTERSECT(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 p1 ← next(p1)
 9                else p2 ← next(p2)
10  return answer
```

- if the list lengths are x and y, the merge takes O(x+y) operations
- Crucial : **postings sorted by docID**

  → could do a linear scan through the two postings lists

  정렬을 하지 않는 경우에는 최악의 경우 $x \times y$번 연산

출처 : 스탠포드 IR 강의 (https://www.youtube.com/watch?v=5KbynCj7yRQ&list=PLaZQkZp6WhWwoDuD6pQCmgVyDbUWl_ZUi&index=4)