# Exercise 1

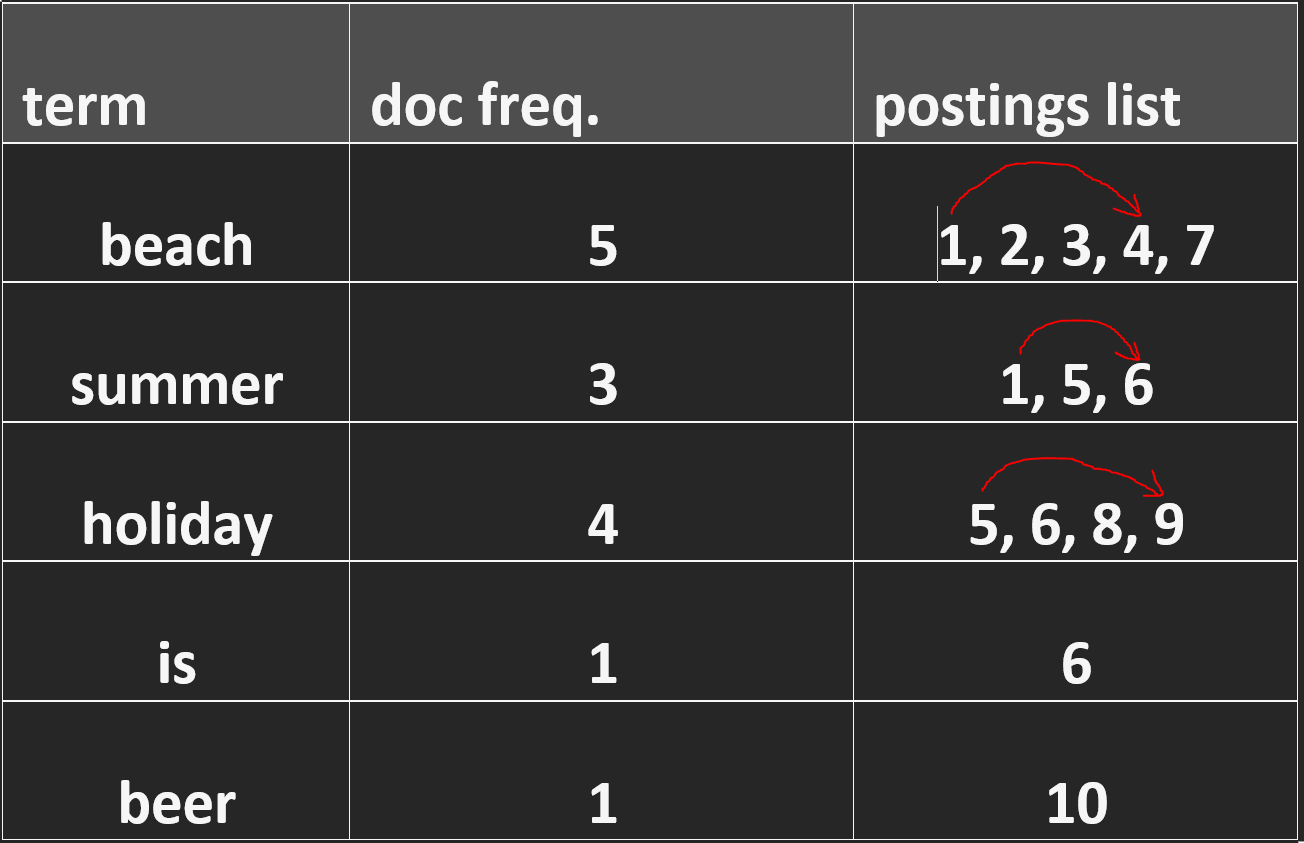
# Task 1

## Subtask A

|  |  |  |
| --- | --- | --- |
| **term** | **doc freq.** | **postings list** |
| **beach** | **5** | **1, 2, 3, 4, 7** |
| **summer** | **3** | **1, 5, 6** |
| **holiday** | **4** | **5, 6, 8, 9** |
| **is** | **1** | **6** |
| **beer** | **1** | **10** |

## Subtask B

Skip pointers added:



Example:

* Add skip pointer for postings list for the term beach from 1 to 4
* The query:
  + **beach AND holiday**
* We start at the postings lists with the entries **beach = 1** and **holiday = 5**
* So we can take the skip pointer at the term beach from 1 to 4, so we are at beach = 4
* Next we compare beach = 7 🡪 Now we know that there is no match
* **The skip pointer saved us the comparisons ob 1,2,3 for the term beach.**
  + Without skip pointers we need more comparisons, thus this query can be answered in a more efficient way with these skip pointers.

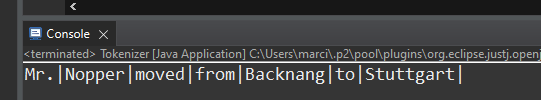
# Task 2

## Pseudo-Code:

1. Initialize currentChar = 0
2. Initialize lastStop = 0
3. Initialize resultList = []
4. Iterate over inputString at position currentChar with condition currentChar < inputString.length
   1. If inputString[currentChar] is Whitespace add substring of inputString starting from index lastStop, ending at index currentChar to resultList. Set lastStop = currentChar+1
   2. If currentChar equals inputString.length – 1 add substring of inputString starting from index lastStop, ending at index currentChar+1 to resultList.
   3. currentChar++ and continue iteration at 4
5. return resultList

## Java-Code implementation:

1. **public class Tokenizer {**
2. **public** **static** String example = "Mr. Nopper moved from Backnang to Stuttgart";
4. **static** List<String> tokenize(String input) {
5. **List<String> result = new ArrayList();**
7. **int** lastStop = 0;
9. **for**(**int** currentChar=0; currentChar<input.length(); currentChar++) {
10. **if(Character.isWhitespace(input.charAt(currentChar))) {**
11. result.add(input.substring(lastStop, currentChar));
12. lastStop = currentChar+1;
13. }
14. **if**(currentChar == input.length()-1) {
15. **result.add(input.substring(lastStop, currentChar+1));**
16. }
17. }
18. **return** result;
19. }
21. **public** **static** **void** main(String[] args) {
22. List<String> test = tokenize(example);
23. test.**forEach**(t -> System.out.print(t+ "|"));
24. }
25. **}**

Output:

The presented algorithm has in general a linear runtime, because it only iterates the inputString once with the for-Loop and the loop variable currentChar is never changed/reset inside the loop. Therefore the for-loop is executing exactly n-times for an inputString of length n. All other operations (variable initialization) have a runtime of O(1). This results in a O(n) runtime for the algorithm.