*Alberto Saponaro - saponaroalberto97@gmail.com*
*Walter Väth - walter.vaeth@gmail.com*
*Chong Shen - st143575@stud.uni-stuttgart.de*
*Xin Pang - Email*

# Pen and Paper Task 1

## Subtask A

- sun → 1, 2, 3, 4, 7
- nice → 1, 5, 6
- water → 5, 6, 8, 9
- is → 6
- beer → 10

## Subtask B

- sun → 1, 2, 3, 4, 7
- nice → 1, 5, 6
- water → 5, 6, 8, 9
- is → 6
- beer → 10

Example query: *nice AND is*

1. Comparisons without skip pointers: 1 & 6, 5 & 6, 6 & 6 ⇒ 3 comparisons
2. Comparisons with skip pointers: 1 & 6, 6 & 6 ⇒ 2 comparisons

Without skip pointers we must compare the terms step by step, although 5 in **nice** is still smaller than 6 in **is**. With skip pointers we can skip the 5 in **nice** and directly go to the 6 in **nice**.

*Alberto Saponaro - saponaroalberto97@gmail.com*
*Walter Väth - walter.vaeth@gmail.com*
*Chong Shen - st143575@stud.uni-stuttgart.de*
*Xin Pang - Email*

## Pen and Paper Task 2

```
    tokenize(text: string):
        token = ''
        list_of_tokens = []

        for char in text:
            if char == ' ':  #whitespace
                list_of_tokens.add(token)
                token = '' #empty string
            if char is a symbol:
                list_of_tokens.add(token)
                list_of_token.add(char)
                token = '' #empty string
            else:
                token += char

        return list_of_tokens
```

*Alberto Saponaro - saponaroalberto97@gmail.com*
*Walter Väth - walter.vaeth@gmail.com*
*Chong Shen - st143575@stud.uni-stuttgart.de*
*Xin Pang - Email*

## Pen and Paper Task 3

Query: *Gates /2 Microsoft*

(Gates, 4): [{1:[3], 2:[6], 3:[2, 17], 4:[1]}]
(Microsoft, 4): [{1:[1], 2:[1, 21], 3:[3], 5:[16, 22, 51]}]

cross product = { 1:[(3,1)], 2:[(6,1), (6,21)], 3:[(2,3), (17,3)], 4:[(1,16), (1,22), (1,51)] }

From all tuples in the cross product, the tuples (3,1) and (2,3) fulfill the query's condition abs(tuple[1]-tuple[0]) $\leq 2$. So the answer is: document 1, document 3.

*Alberto Saponaro - saponaroalberto97@gmail.com*
*Walter Väth - walter.vaeth@gmail.com*
*Chong Shen - st143575@stud.uni-stuttgart.de*
*Xin Pang - Email*

## Programming Task 1

```
import csv, re, nltk

def index(filename: str ='IRTM/assignment1/code/postillon.csv'):
    index = {}
    dictionary = {}
    postings_lists = []

    tokenizer = nltk.RegexpTokenizer(r"\w+")

    with open(filename, 'r') as file:
        reader = csv.reader(file, delimiter = '\t')
        postings = []

        #iterate through each row of the table
        for row in reader:
            (doc_id, url, pub_date, title, news_text) = row

            #tokenize and normalize news text
            #this procedure will remove symbols like !?() etc.
            #the set data structure will remove all duplicates
            news_text_norm = set(tokenizer.tokenize(news_text.lower()))

            #generate postings
            #iterate through each term
            for term in news_text_norm:
                postings.append((term, doc_id))

        #sort postings
        postings = sorted(postings[1:], key = lambda tup: tup[0])


        post_id = 0
        post_size = 0
        #iterate through postings
        for posting in postings:
            term, doc_id = posting

            if term not in dictionary:
                #upate the dictionary with the new term
                #initialize the postings size
                #save the postings id,
                #witch is the position of the postings list
                #into the postings lists
                dictionary.update({term: [post_size+1, post_id]})

                #initialize a new postings list
                postings_lists.append([doc_id])

                #update postings id
                post_id +=1
            else:
                #update size of posting
                dictionary[term][0] += 1
```

Alberto Saponaro - saponaroalberto97@gmail.com
Walter Väth - walter.vaeth@gmail.com
Chong Shen - st143575@stud.uni-stuttgart.de
Xin Pang - Email

```
54
55                  #update postings list
56                  postings_lists[-1].append(doc_id)
57
58      return dictionary, postings_lists
59
60
61  def query(data: tuple, term_1: str, term_2: str = ''):
62      dictionary, postings_lists = data
63      intersect = []
64      post_size = 0
65      post_id = 0
66
67      #CASE 1: the query contaiuns only one term
68      if term_2 == '':
69          #iterate through terms in dictionary
70          for term in dictionary:
71              if term_1 == term:
72                  post_size, post_id = dictionary[term]
73                  break
74
75          #set the index of the first postings list
76          idx = 0
77          #iterate through postings lists
78          for postings_list in postings_lists:
79              if post_id == idx:
80                  return postings_list
81                  break
82              else:
83                  #update index
84                  idx += 1
85
86      #CASE 2: the query contains two terms
87      else:
88          #term_1 AND term_2
89          term_1_post_id = dictionary[term_1][1]
90          term_2_post_id = dictionary[term_2][1]
91
92          #set the index of the first postings list
93          idx = 0
94          #iterate through postings lists
95          for postings_list_1 in postings_lists:
96              if term_1_post_id == idx:
97                  return postings_list_1
98                  break
99              else:
100                 #update index
101                 idx += 1
102
103         #set the index of the first postings list
104         idx = 0
105         #iterate through postings lists
106         for postings_list_2 in postings_lists:
107             if term_2_post_id == idx:
108                 return postings_list_2
109                 break
```

Alberto Saponaro - saponaroalberto97@gmail.com
Walter Väth - walter.vaeth@gmail.com
Chong Shen - st143575@stud.uni-stuttgart.de
Xin Pang - Email

```
110             else:
111                 #update index
112                 idx += 1
113 
114         for doc_id_1 in postings_list_1:
115             for doc_id_2 in postings_list_2:
116                 if doc_id_1 == doc_id_2:
117                     intersect.append(doc_id_1)
118                 if doc_id_1 > doc_id_2:
119                     break
120 
121 
122         return intersect
123 
124 
125 
126 if __name__ == "__main__":
127     data = index()
128     print(len(query(data, 'wei ', 'ma ')))
129     print(len(query(data, 'wei ', 'masse')))
130     print(len(query(data, 'weiss', 'ma e')))
131     print(len(query(data, 'weiss', 'masse')))
```

code/script.py