

Pen and Paper Task 1

Subtask A

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

Subtask B

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

Example query: *nice AND is*

1. Comparisons without skip pointers: 1 & 6, 5 & 6, 6 & 6 \Rightarrow 3 comparisons
2. Comparisons with skip pointers: 1 & 6, 6 & 6 \Rightarrow 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**.

Pen and Paper Task 2

Prp 2)

```

to kenize(text) {
    list = []
    begin = 0
    end = 0
    current = 0

    while (current < text.length) {
        while (text[current+1] ∈ {0, 1, 2, A, 3, 0, 9}) {
            current++
        }
        end = current + 1
        if (text[begin:current] is not in list)
            list.add(text[begin:current])
        begin = current + 1
        current++
    }
}

```

$\Theta(n)$: each element of the text will be checked of validity by using the index "current" for exact one time, that leads to the linear run time with respect to the number of the characters in text, namely the input length.

- eg. "0,kennedy,hash,t,gone,to,S."

iter # of complete execution of the outer loop	begin	end	current	list
0	0	0	0	[]
1	2	0	2	[0]
2	10	8	10	[0, kennedy]
3	15	13	15	[0, kennedy, hash]
4	17	15	17	[0, kennedy, hash, t]
5	22	20	22	[0, kennedy, hash, t, gone]
6	25	23	25	[0, kennedy, hash, t, gone, to]
7	27	25	27	[0, kennedy, hash, t, gone, to, S]

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 $\text{abs}(\text{tuple}[1] - \text{tuple}[0]) \leq 2$. So the answer is: document 1, document 3.

Pen and Paper Task 4

P&P4)

$u \backslash v$		w	o	h
	0	1	2	3
w	1	0	1	2
h	2	1	1	1
o	3	2	1	1

To compute one entry $D_{i,j}$ of the table, we have to take the minimum of the following four terms: $D_{i-1,j-1}$ if $u_i = v_j$,
 $D_{i,j-1} + 1$ if $u_i \neq v_j$
 $D_{i,j-1} + 1$
 $D_{i-1,j} + 1$
and $D_{i-2,j-2}$ if $u_i = v_{j-1}$ and $u_{i-1} = v_j$

Programming Task 1

```
1 import csv, re, nltk
2
3 def index(filename: str):
4     index = {}
5     dictionary = {}
6     postings_lists = []
7
8     tokenizer = nltk.RegexpTokenizer(r"\w+")
9
10    with open(filename, 'r') as file:
11        reader = csv.reader(file, delimiter = '\t')
12        postings = []
13
14        #iterate through each row of the table
15        for row in reader:
16            (doc_id, url, pub_date, title, news_text) = row
17
18            #tokenize and normalize news text
19            #this procedure will remove symbols like !?() etc.
20            #the set data structure will remove all duplicates
21            news_text_norm = set(tokenizer.tokenize(news_text.lower()))
22
23            #generate postings
24            #iterate through each term
25            for term in news_text_norm:
26                postings.append((term, doc_id))
27
28            #sort postings
29            postings = sorted(postings[1:], key = lambda tup: tup[0])
30
31
32    post_id = 0
33    post_size = 0
34    #iterate through postings
35    for posting in postings:
36        term, doc_id = posting
37
38        if term not in dictionary:
39            #upate the dictionary with the new term
40            #initialize the postings size
41            #save the postings id,
42            #witch is the position of the postings list
43            #into the postings lists
44            dictionary.update({term: [post_size+1, post_id]})
45
46            #initialize a new postings list
47            postings_lists.append([doc_id])
48
49            #update postings id
50            post_id +=1
51        else:
52            #update size of posting
53            dictionary[term][0] += 1
```

```
54
55         #update postings list
56         postings_lists[-1].append(doc_id)
57
58     return dictionary, postings_lists
59
60 class Search:
61
62     def __init__(self, filename: str, index: tuple):
63         self.filename = filename
64         self.index = index
65
66     def getPostingList(self, postID):
67         postings_lists = self.index[1]
68
69         #set the index of the first postings list
70         idx = 0
71         #iterate through postings lists
72         for postings_list in postings_lists:
73             if postID == idx:
74                 return postings_list
75                 break
76             else:
77                 #update index
78                 idx += 1
79
80     def query(self, term: str):
81         dictionary, postings_lists = self.index
82         postID = dictionary[term][1]
83         postings_list = self.getPostingList(postID)
84
85         #retrive text
86         with open(filename, 'r') as file:
87             reader = csv.reader(file, delimiter = '\t')
88             postings = []
89
90         #iterate through each row of the table
91         for row in reader:
92             (docID, url, pub_date, title, news_text) = row
93             if docID in postings_list:
94                 return(docID, news_text)
95
96     pass
97
98     def query(self, term1: str, term2: str = ''):
99         dictionary, postings_lists = self.index
100         out_list = []
101
102         #CASE 1: only one term
103         if term2 == '':
104             postID = dictionary[term1][1]
105             postings_list = self.getPostingList(postID)
106
107         #retrive text
108         with open(filename, 'r') as file:
109             reader = csv.reader(file, delimiter = '\t')
```

```
110         postings = []
111
112         #iterate through each row of the table
113         for row in reader:
114             (docID, url, pub_date, title, news_text) = row
115             if docID in postings_list:
116                 out_list.append((docID, news_text))
117 #CASE 2: two terms
118 else:
119
120     intersection_list = []
121
122     term1_postID = dictionary[term1][1]
123     term2_postID = dictionary[term2][1]
124
125     term1_postings_list = self.getPostingList(term1_postID)
126     term2_postings_list = self.getPostingList(term2_postID)
127
128     #intersection algorithm
129     for term1_docID in term1_postings_list:
130         for term2_docID in term2_postings_list:
131             if term1_docID == term2_docID :
132                 intersection_list.append(term1_docID)
133
134     #retrive text
135     with open(filename, 'r') as file:
136         reader = csv.reader(file, delimiter = '\t')
137         postings = []
138
139         #iterate through each row of the table
140         for row in reader:
141             (docID, url, pub_date, title, news_text) = row
142             if docID in intersection_list:
143                 out_list.append((docID, news_text))
144     return out_list
145
146
147 if __name__ == "__main__":
148     filename = 'assignment1/code/postillon.csv'
149     index = index(filename=filename)
150     search = Search(filename=filename, index=index)
151
152     #queries
153     print('wei  AND ma ')
154     for item in search.query('wei ', 'ma '):
155         print(item)
156
157     print('wei  AND masse')
158     for item in search.query('wei ', 'masse'):
159         print(item, '\n')
160
161     print('weiss AND ma e')
162     for item in search.query('weiss', 'ma e'):
163         print(item, '\n')
164
165     print('weiss AND masse')
```

```
166 for item in search.query('weiss', 'masse'):  
167     print(item, '\n')
```

code/script.py

The result shows the docID with the news text. With the use of an iterator -in Python simply a for loop- I iterate through the first and second postings list to find if the two terms are found in both documents.

The differences between the two queries is that in the first one we don't need an intersect algorithm. In this case we can use the postings list of the given term directly. In the second one with two terms we have to calculate the intersection of the two postings lists and the return the text of the documents that have both terms in it.

Terminal OUTPUT:

```
1 weiß AND maß  
2 ('691', '    +++ W  rde am liebsten im Erdboden versinken: Maulwurf hat das GraACben  
    verlernt +++ +++ Aus Fluss: Schwimmerin wei  , woher sie Entz  ndungskrankheit hat  
    +++ +++ Es ist ein M  dchen!: Fliege bekommt Nachwuchs +++ +++ Nur an der Nase  
    herumgef  hrt: Masochist unzufrieden mit Domina-Dienstleistung +++ +++ Ma   Capone  
    : Schneider von Gangsterboss erh  lt Frischk  se zum Dank +++ +++ Hat Eindruck  
    hinterlassen: J  rgen W. M  llemann +++ +++ Stehen auf der Gersteliste: Gerd Reide  
    und Johann Hafer zu Roggen-Roll-Party eingeladen +++    Jetzt bestellen! Die  
    besten Newsticker als Buch (nur 9,9  Q: Der Postillon: +++ Newsticker +++ ')  
3  
4 wei   AND masse  
5 ('864', '    Berlin (dpo) - Die Polizei spricht von einem gef  hrlichen neuen Drogen-  
    Trend: Immer mehr Berliner begeben sich seit gestern mit Bananenresten in den  
    Atemwegen in   rztliche Behandlung. Intensive Recherchen des Postillon im  
    Berliner Drogenmilieu haben ergeben, dass der Ursprung der mysteri  sen F  lle  
    offenbar bei einer 140-Kilo-Lieferung eines neuen Mode-Kokains aus Kolumbien  
    liegt, die seit Dienstagvormittag im Umlauf ist. Wie der Stoff in die Szene  
    geraten ist, bleibt unklar. Nach Angaben von Dealer Henrik K.* (*Name von der  
    Redaktion ge  ndert), der mehrere Kilogramm des neuartigen Kokains kaufte, hat  
    die Lieferung gravierende M  ngel. "Nicht einmal gerade Lines lassen sich mit dem  
    Zeug legen", klagt der 32-J  hrige. Ist unzufrieden mit dem neuen Stoff:  
    Henrik K. Einige seiner Kunden, die spekulierten, dass es sich bei der neuen,  
    gelb-gebogenen Darreichungsform um Verpackung handeln k  nnte und nach  
    stundenlangem Kampf zum Inneren vordrangen, berichten von einer matschigen Masse  
    , die zwar einigerma  en wei  , aber kein bisschen feink  rnig sei und nur die Nase  
    verstopfe. Lediglich ein Abnehmer, der das neue Kokain im LSD-Rausch a  , habe  
    von einem wohligen V  llegef  hl berichtet und wolle unbedingt mehr von dem "  
    Wunderzeug" kaufen. Henrik K. sch  pft daraus jedoch wenig Trost: Seit gestern  
    trocknet er 3 Kilogramm der Masse in seinem Backofen, um sie anschlie  end zu  
    zermahlen und doch noch irgendwie nutzbar zu machen. "Wenn das nicht klappt,  
    kann ich nur noch schauen, ob es vielleicht intraven  s funktioniert. Sonst wei    
    ich auch nicht weiter." Zwar scheint eine   berdosis mit dem neuen Stoff nicht m    
    glich, acht Konsumenten werden aber wohl bleibende Sch  den davontragen. Sie  
    leierten ihre Nasenl  cher beim Versuch, das Kokain ungesch  lt zu schnupfen,  
    hoffnungslos aus. ')  
6  
7 ('2197', '    Osnabr  ck (dpo) - M  ssen Fans bald auf eine liebgewonnene Tradition  
    verzichten? Nachdem Schiedsrichter Martin Petersen beim Pokalspiel zwischen dem  
    VfL Osnabr  ck und RB Leipzig von einem Feuerzeug getroffen wurde, erw  gt der DFB
```


nun ernsthaft, das Werfen von Feuerzeugen in Fu  ballstadien unter Verbot zu stellen. Man bef  rchtet, dass von den Projektilen Verletzungsgefahr ausgeht. "Fu  ball ist ein Erlebnis f  r die ganze Familie. Deshalb spielen wir mit dem Gedanken, Feuerzeugw  rfe – so sch  n sie auch anzuschauen sind – zu untersagen", erkl  rte DFB-Pr  sident Wolfgang Niersbach. "Derzeit werben wir f  r die Zustimmung der Vereine zu dieser sicherlich umstrittenen Ma  nahme." Darf bald nur noch zum Entz  nden von Pyrotechnik verwendet werden: Feuerzeug Fan-Initiativen zeigen sich emp  rt: "Fu  ball lebt von Emotionen", erkl  rt etwa Ren   Finkel vom Osnabr  ck-Fanclub Lila-Wei   Westerkappeln. "Da geh  rt das Werfen von Feuerzeugen aus einer anonymen Masse einfach dazu. Was kommt als N  chstes? Ein Verbot, den Platz zu st  rmen?" Tats  chlich hat der Feuerzeugwurf Tradition. Er geht zur  ck bis Mitte des 19. Jahrhunderts, als es   blich war, Feuersteine und Zunderdosen auf das Spielfeld zu werfen. Ziel der W  rfe war es schon damals, gegnerische Spieler oder den Schiedsrichter m  glichst schwer zu verletzen und einen Spielabbruch herbeizuf  hren. W  hrend der Weimarer Republik geriet der Brauch nahezu in Vergessenheit. Zu wenig Wucht entwickelten die damals   blichen Streichh  lzer. Erst mit der massenhaften Einf  hrung des Feuerzeugs erlebte der Z  ndmittelwurf eine Renaissance. Sollte das Feuerzeugwurfverbot tats  chlich kommen, m  ssten zahlreiche Fanclubs und Ultra-Gruppierungen ihre Feuerzeugwurf-Bataillone aufl  sen oder auf andere Wurfgegenst  nde umlernen lassen. Auch die beliebten Wurffeuerzeug-St  nde in den Stadien w  rden dann der Vergangenheit angeh  ren. ')

('4933', ' K  ln, Stuttgart, M  nchen (dpo) – Ein meteorologisches R  tsel sorgt derzeit in weiten Teilen Deutschlands f  r gro  e Verunsicherung. Der Grund: In vielen Regionen fiel im Laufe des Tages eine bislang noch unbekannte nasskalte, helle Masse vom Himmel. Vielerorts blieb die Substanz liegen und f  rbte die Landschaft leuchtend wei   ein. Laut Beh  rden besteht keine Gefahr f  r die Bev  lkerung. "Es war unheimlich", berichtet Hausfrau Renate P. aus Stuttgart dem Postillon. "Pl  tzlich hab ich was Kaltes auf der Nase gesp  rt. Ich schau nach oben und seh, dass da so wei  es Zeug runterrieselt. Ganz leise. Sah aus wie bei mir daheim in der Tiefk  hltruhe." Frau P. fl  chtete sich daraufhin gemeinsam mit anderen Passanten in einen Laden, wo sie mehrere Stunden lang ausharrte. "Maaammiiiiiii! Ich hab Angst!" Inzwischen haben Meteorologen eine Erkl  rung f  r das Ph  nomen parat. Demnach soll es sich nicht wie zun  chst vermutet um Vulkanasche oder radioaktiven Fallout handeln. "Vielmehr haben wir es bei der wei  en Substanz mit einer speziellen Form von Niederschlag zu tun, wie man sie sonst aus dem skandinavischen Raum oder aus alten Erz  hlungen kennt", erkl  rt Susanne Brauer vom Deutschen Wetterdienst. "Fr  her hat diese Substanz in der Winterzeit oft weite Teile Europas bedeckt." Zwar halten die Beh  rden den Niederschlag bislang f  r ungiftig, doch zumindest auf Autofahrer scheint die unbekannte Masse einen bizarren Effekt zu haben. Viele von ihnen reagieren panisch und verwirrt und scheinen innerhalb k  rzester Zeit selbst die grundlegendsten Regeln des Stra  enverkehrs vollst  ndig zu vergessen. Vorerst sollen deshalb die kommunalen Winterdienste s  mtliche Stra  en mit extrem langsam fahrenden R  um- und Streufahrzeugen blockieren, bis sich die Lage wieder beruhigt hat. ')

weiss AND ma  e
weiss AND masse