Due: 2018/3/16

Homework 2

Name:

Problem 1. (20 points) Below is a portion of a positional index in the format:
term: doc1: <position1, position2, ...>; doc2: <position1, position2, ...>; etc

angles: 2 < 36, 174, 252, 651 >; 4: < 12, 22, 102, 432 >; 7: < 17 >
fools: 2: <1,17,74,222>; 4: <8,78,108,458>; 7: <3,13,23,193>;
fear: 2: <87,704,722,901>; 4: <13,43,113,433>; 7: <18,328,528>;
in: 2: <3,37,76,444,851>; 4: <10,20,110,470,500>; 7: <5,15,25,195>;
rush: 2: <2,66,194,321,702>; 4: < 9,69,149,429,569>; 7: <4,14,404>;
to: 2: <47,86,234,999>; 4: < 14,24,774,944>; 7: <199,319,599,709>;
tread: 2: <57,94,333>; 4: <15,35,155>; 7: <20,320>;
where: 2: <67,124,393,1001>; 4: <11,41,101,421,431>; 7: <16,36,736>;

Which document(s) if any meet each of the following queries, where each expression with in quotes is a phrase query?

a. "fools rush in "

b. "fools rush in" *AND* "angels fear to tread"

Solution:
(a) "fools rush in"
All three documents (2,4 and 7) satisfy the query. (2 points for the right answer, 1 points for partially correct answer such as 2 and 4).
(b) "fools rush in" AND "angels fear to tread"
Only 4 satisfies.

Problem 2. (30 points)

a. Write down the entries in the permuterm index dictionary that are generated by the term *conflict*.

b. Consider the query *conf*ct*, what Boolean query on a bigram index would be generated for this query?

c. Can you think of a term that satisfies the Boolean query in question b. but does not match the permuterm query *ct$conf**? What about the reverse case?

Solution: **(a)** The term "conflict" generates the following permuterm index dictionary :
conflict$*, *onflict$c, *nflit $co*, *flict$con*,
lict$conf, *ict$confl*,
ct$confli, *t$conflic*, *$conflict*

(b) Query **conf$*$ct** lookup on **ct$*$conf**. The following boolean query on a bigram index will be generated.
ct **AND** t\$**AND** \$c **AND** co **AND** on **AND** nf
(c) The following queries satisfies the boolean query in b, but doesn't match the permuterm query **ct \$conf***
"act confident"
For this reverse case, there is no such query that matches the permutation query but does not satisfy the above boolean query because no matter what we replace in *, we would always have the bigrams of boolean in that term.

Problem 3. (30 points) Given two strings S_1 and S_2, write down the pseudo-code of computing the edit distance between them.
Solution:

Algorithm 1: EDIT_DISTANCE($S_1[1..M]$, $S_2[1..N]$)

```
$D(i, j)$
$D(i, 0) \leftarrow i$
$D(0, j) \leftarrow j$
for $i \leftarrow 1$ to $M$ do
    for $j \leftarrow 1$ to $N$ do
        $D(i, j) = min($
                    $D(i-1, j) + 1,$
                    $D(i, j-1) + 1,$
                    $D(i-1, j-1)+$ 2; if $S_1[i] \neq S_2[j]$
                                   0; if $S_1[i] = S_2[j]$
                   );
return $D(N, M)$
```

Problem 4. (20 points) Write the pseudo code showing the details of computing the Jaccard coefficient while scanning the posting of the k-gram index. (Page 49 in the slide)
Solution:

Algorithm 2: Psuedocode: Coefficient of K-gram Index)

```
$Jaccard(query, t)$
$A$ =Set of k-grams of t
$B$ = Set of k-grams of query
$count = 0$
for $i \leftarrow 1$ to $length(B)$ do
    $list$ = postingList of $B[i]$
    $if(list.CONTAINS(t))$
    $count++;$
    $Jaccard$ co-efficient = $\frac{count}{(length(A)+length(B)-count)}$
```