COMP6714 ASSIGNMENT 1

DUE ON 20:59 4 NOV, 2020 (WED)

Some Boolean retrieval syst /S, which restricts the occurrences matches to be within the same sentence.

Assume that we have created an additional positional list for `$`, which records the positions of the end of the sentences. E.g. for the document

```
A B C. D E.
```

the position list for `$` is `[4, 7]`.

You are required to engine an algorithm to support the query sk easier, we further constrain the search results of the query to satisfy s:

- the occurrences of `A` and `B` must be within the same sentence.
- the occurrence o

For example, the above ex not C /S A.

You need to

- make simple modifications to the pseudocode shown in Alg actly the algorithm in Figure 2.12 in the textbook. Note that we mod algorithm slightly so that array indexes start from 1 instea
  - you need to insert some code between Lines 6 and 7, and p ifications to some lines afterwards.
  - In your submitted algorithm pseudocode (named $\mathsf{Q1}(p_1, p_2, p_\$)$), clearly mark the modifications using color or boxes.
- You can assume that there is a function $\mathsf{skipTo}(p, docID, pos)$, which move the cursor of list p to the first position such that (1) the position belongs to a document $docID$, and (2) the position is no smaller than pos.

Q2. (*25 marks*)

Consider the scenario of dynamic inverted index construction. Assume that t sub-indexes (each of M pages) will be created if one chooses the no-merge strategy.

(1) Show that if the logarithmic merge strategy is used, it will result in at most $\lceil \log_2 t \rceil$ sub-indexes.
(2) Prove that the total I/O cost of the logarithmic merge is $O(t \cdot M \cdot \log_2 t)$.

Algorithm 1: PositionalIntersect(p_1, p_2, k)

```
1  answer ← ∅;
2  while p1 ≠ nil ∧ p2 ≠ nil do
3      if docID(p1) = docID(p2) then
4          l ← [ ];
5          pp1 ← positions(p1); pp2 ← positions(p2);
6          while pp1 ≠ nil do
7              while pp2 ≠ nil do
8                  if  pos(pp )    pos(pp )     k then
9                      
10                 el
11                     
12                         break;
13                 pp2 ← next(pp2);
14             while l ≠ [ ] ∧ |l[1] − pos(pp1)| > k do
15                 delete(l[1]);
16             for each ps ∈ l do
17                 answer ← answer ∪ {docID(p1), pos(p1), ps}
18             pp1 ← next(pp1)
19         p1
20     else
21         if d
22             
23         else
24             p2 ← next(p2);
25 return answer;
```


Q3. (*25 marks*)

After the δ encoding, the compressed non-positional inverted list is

```
01000101  11110001  01110000  00110000  11110110  11011
```

- Decode the sequence of numbers the compressed list represents.
- List the document IDs in this list.

Q4. (*25 marks*)

Consider the WAND algorithm described in Section 2.4 in the original paper.[1]. There is a typo in the algorithm in Line 21: it should use "terms[0 .. (pTerm-1)]".

[1] Efficient Query Evaluation using a Two-Level Retrieval Process.

However, even with this fixed, there is a bug in the algorithm (Figure 2) in which the algorithm will end up in an infinite loop.

You need to

- Identify the **single** lines in Figure 2 that causes the bug and describe concisely why this will lead to a bug.
- Give a simple example illustrating this bug. You should use three terms (named A, B, C) and $k = 1$. Do not include unncessary entries in the lists.

	A	B	C
U			
List			

Submission Instructions

You need to write your solution to the questions in a pdf file named `1.pdf`. You **must**

- include your
- the file can be open

You need to show the key step

Note: Collaboration is allowed. However, each person must independently write up his/her own solution.

You can then submit the file by `give cs6714_ass1 ass1.pdf` to 5MB.

Late Penalty: **-10%** per day for the first two days, and **-20%** per day for the following days.