COMP6714 (13S2) MID-TERM EXAM

TIME ALLOWED: 1 HOUR

Name: _____

Student ID: _____

NOTE:

(1) Answer the questions briefly. Lengthy but irrelevant answers will be penalized.

(2) In most of the questions, you need to show major steps.

Q1. (*30 marks*)

Answer the following questions.

(1) What is the heuristic method to determine the execution order using the binary list merge algorithm to answer *conjunctive* keyword queries? Given a counter-example where this heuristic does not work well.

Your Answer:

The execution order should follow with an increasing order of the inverted index list of query.

For example, there is a query A and B and C where len(list(A)) < len(list(B)) < len(list(C)) According the heuristic method, we should calculate it as (A and B) and C.

But if len(A and B) = len(A), while the result of it have common only 1 sample with C, then the heuristic doesn't work well

(2) Given at least two reasons (with simple examples) why language identification is important when indexing documents.
Your Answer:

① In some language, there is no space - such as Chinese and Japanese so splitting word in these two is harder.
② Some language may have different writing order for example, Arabic write from right to left, while numbers wrote from left to right
③ In German, needs compounts splitter.

(3) Why specialized algorithms are needed to construct inverted index for large document collections?
Your Answer:

The collections may be extremely large, so we can't generate the inverted index just in memory, some external methods are needed.

(4) What is the Heaps' Law and what is the Zipf's Law?
Your Answer:

Heaps' Law = $M = kT^b$, where M is the size of vocabulary and T is the size of collection.

Zipf's Law: The ith most frequent words has frequency proportional to $1/i$. $cf_i \propto 1/i$

(5) Give an intuitive explanation of the *two* major factors we consider when choosing the best candidate for *context-sensitive* spelling correction. (For example, `taw` may be corrected to either `the` or `thaw`)
Your Answer:

?

Q2. (*20 marks*)

Complete the pseudocode of the function Q2 (shown below) that answers the Boolean keyword query "*A* AND (NOT *B*)", where *A* and *B* are two different keywords. Make sure your pseudocode is easily readable (i.e., with indentation and comments if necessary).

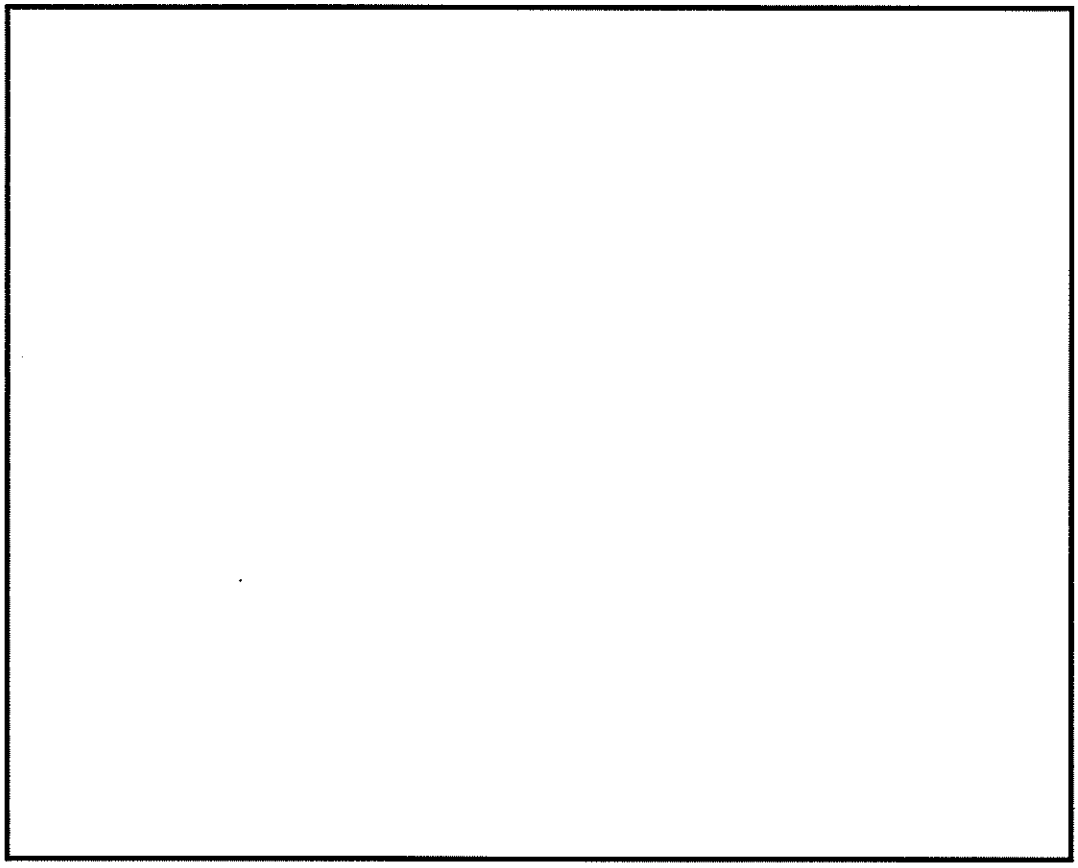
You can assume the following functions/methods on LA or LB:

- cur() returns the current docID in the list;
- eol() returns TRUE if the current list is exhausted;
- next() moves the cursor to the next posting.

```
function Q2(LA, LB)
  // Let LA and LB be the corresponding inverted lists
  // of A and B, respectively
  ANSWER = []
  ...
  ...
  return ANSWER
end
```

Your Answer:

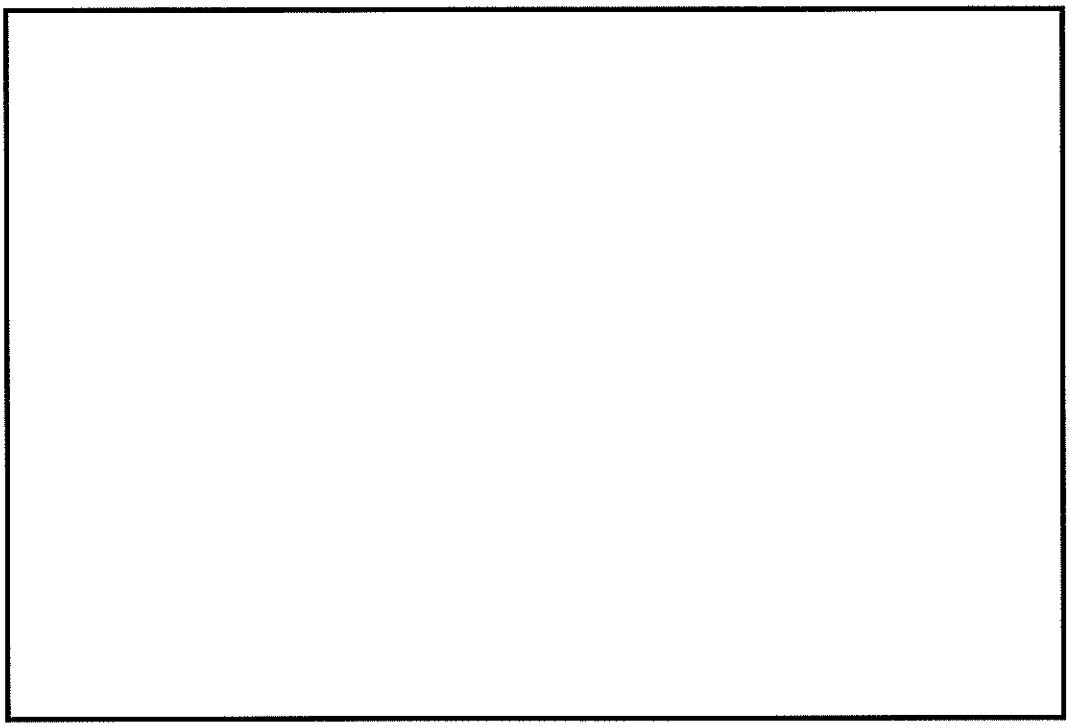
```
function Q2( LA , LB)
    ANSWER = []
    pA -> LA
    pB -> LB
    while (pA != NULL and pB != NULL)
        if ( cur(pA) = cur(pB) ) then
            pA = next(pA)
            pB = next(pB)
        else if cur(pA) > cur(pB) then
            p(B) = next(pB)
        else if cur(pA) < cur(pB) then
            ADD(answers, cur(pA))
            p(A) = next(pA)
    end if
    end while
    if eol(pB) then
        while ! eol(pA) do
            add(answers, cur(pA))
            p(A) = next(pA)
        end while
    end if
    return answers.
```



Q3. (*25 marks*)

Given a list of sorted strings of possibly different length, describe a method to use binary search to answer prefix query P*, where P is a non-empty string (You may use an additional $O(n)$ space for some auxiliary data structures)? Why B^+-tree index is still preferred over this binary-search-based solution for large collection of strings?

Your Answer:



Q4. (*25 marks*)

Consider using one of the three dynamic indexing methods, namely *immediate merge*, *no merge*, and *logarithmic merge*, to build the inverted index for a collection. Let $|C|$ be the total number of bytes of the documents in the collection, M be the memory size in bytes, and B be the number of bytes in a disk block. $\lfloor \frac{M}{B} \rfloor$ blocks

We only consider the total number of blocks read from or written to the disk as the I/O cost; and you can safely ignore the ceiling or floor functions in the analysis. For example, reading 1000 bytes from the disk costs $\frac{1000}{b}$ I/Os.

(1) How many sub-indexes will the *no merge* method create? What is the total I/O cost of indexing the collection?

(2) How many sub-indexes will the *immediate merge* method create? What is the total I/O cost of indexing the collection?

(3) How many sub-indexes will the *logarithm merge* method create (you may consider the worst case)? What is the total I/O cost of indexing the collection?

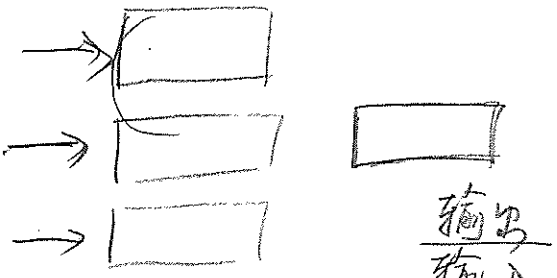
Your Answer:

(1) $\frac{|C|}{M}$ sub-indexes

No-merge first read M bytes, then do in memory sort, then output it.

So there are $\frac{|C|}{M}$ batches. hence $\frac{|C|}{M}$ sub-indexes. I/O cost for each batch is 2 for reading and writing. so the total I/O cost is $2 \cdot \frac{|C|}{M} \cdot \frac{M}{B} = \frac{2|C|}{B}$

(2)



入 K 次也即 K 个 batch，$K=\frac{|G|}{M}$

第1个 batch 开销：$2\cdot\frac{M}{B}$

第2个 batch 开销：$2\cdot\frac{M}{B}+\left(\frac{M}{B}+\frac{M}{B}\right)\times 2$

第3个 batch：$2\cdot\frac{M}{B}+\left(\frac{2M}{B}+\frac{M}{B}\right)\times 2$

⋮

第K个 batch：$2\cdot\frac{M}{B}+\left(\frac{(K-1)M}{B}+\frac{M}{B}\right)\times 2$

∴ Total 开销 $=2K\cdot\frac{M}{B}+\sum_{i=2}^{K}\left(\frac{(i-1)\cdot M}{B}+\frac{M}{B}\right)\times 2$

$$=2K\cdot\frac{M}{B}+\sum_{i=2}^{K}\left(\frac{iM}{B}\right)\cdot 2$$

$$=2K\cdot\frac{M}{B}+\frac{2M}{B}\cdot\sum_{i=2}^{K}(i)$$