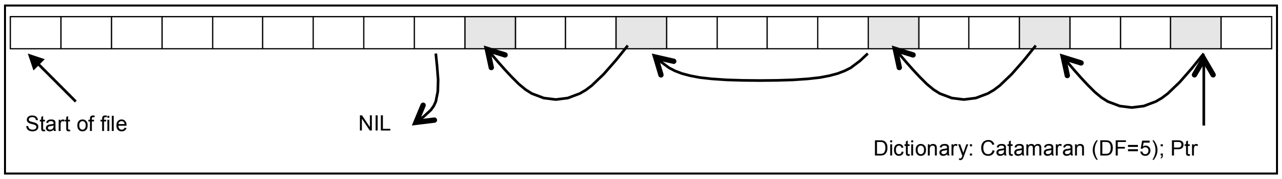
Problem 1 (15 points)

Prof. Quick Li has proposed a new algorithm for index construction. She claims that her method: (1) runs in linear time; (2) requires minimal computer memory (just enough to store the dictionary and the current document), and (3) the algorithm does not require any sorting or merging. Her method works like this. Fixed size records are used to store each <docid, count> pair along with a pointer to the previously written record for the given term; if this is the first time the term is encountered, a special NIL pointer is stored instead. Each record could be represented as three integers. For each term, the dictionary stores document frequency and a pointer (i.e., the pointer is a file offset) to the last observed record. As each term in a document is processed a new record is written on disk and it points to the previous record location stored in the dictionary, which is then updated to point to the current record. One way of viewing this is that the index consists of interwoven linked lists for each term, as in the figure below.



For this problem, assume that we are not concerned about index compression.

(a) Which of the professor's three claims is true?
(b) Would an index built by this method provide enough information to support vector cosine querying with TF/IDF weights?
(c) Another professor says that there is a significant problem using this representation. Is this true? Explain.
(d) An advantage of this approach is that it seamlessly supports collections where new documents will be added over time. Can you think of any way to improve an index that was built in this fashion?

Problem 2 (20 points)

(a) Rank documents D1, D2, and D3 from the following term-document matrix against query Q using a unigram statistical language model. The query contains the words "aardvark bird egret egret". The corpus consists of only these eight documents. For your convenience, collection frequencies, document frequencies, and IDF values are provided for each word. For smoothing purposes you should use a mixture model with $\lambda = 0.5$. You can assume that the prior probability of relevance is the same for each document.

	D1	D2	D3	D4	D5	D6	D7	D8	Q
aardvark	1	4		2			3		1
bird		2	4						1
cat		1		1					
dog	3		1	1			5		
egret			1						2
fish	1	2	1	1	1	1	6	8	

Words	CF	DF	IDF
aardvark	10	4	1
bird	6	2	2
cat	2	2	2
dog	10	4	1
egret	1	1	3
fish	21	8	0

(b) Unlike vector cosine models with TF/IDF weighting, statistical language models do not explicitly weight terms with IDF. Yet language models appear to be more effective. Explain why IDF weighting is not needed.

Problem 3 (15 points)

Inverted file compression can reduce I/O bandwidth and reduce disk space requirements. Given the following term occurrence information, show how gamma compression can be used to reduce the size of postings lists in an inverted file. For this problem we'll use the terms 'sail' and 'boat' (and 'wind').

Term	DF	Docids-TermCount Pairs
sail	3	{<5,1>, <20, 1>, <148, 2>}
boat	2	{<5,1>, <148, 1>}

(a) Assuming docids must be encoded as 4-byte integers and term counts are represented as 1-byte integers, compute the storage required for the two terms without use of compression.

(b) Next, show how the lists of docids and term counts can be represented using gamma compression. You should create a representation, in bits, and please label your work.

(c) What savings, if any, is obtained when gamma compression is used?

(d) Now suppose that term 'wind' occurs in exactly 3 documents and the inverted file contains this bit-string: "1111001111110010110110" which represents the gap list for the 3 documents it occurs in. Decode the bit-string and determine which 3 documents (i.e., docids) contain the word 'wind'. Note, the string only stores document information, not term counts.

Problem 4 (10 points)

Suppose a query has 8 relevant documents: D7, D18, D21, D39, D51, D54, D67, and D79, and assume that all other documents are not relevant. Two retrieval systems Bang and Gaggle retrieve the following ranked lists. (Note: D18 is the 1st ranked doc by Bang; D67 is its 2nd ranked doc, etc...)

Bang: D18, D67, D3, D54, D37, D21, D99, D82, D1, D5, D91, D34

Gaggle: D3, D1, D94, D14, D20, D54, D18, D7, D79, D39, D99, D67

(a) For both systems what is P@10 (precision at 10 documents) for this query?
(b) For both systems what is average precision?
(c) Based on this query, which system would be better for a legal case-law search application?

Problem 5 (8 points)

(a) Front-coding is one way to reduce the size of a dictionary by representing word entries in a compact fashion. For the small set of words below, demonstrate how they may be more efficiently represented using front-coding and quantify the savings.

(b) Your classmate Anna has suggested that front-coding might be more effective if strings are first reversed (e.g., ‘words’ is stored as ‘sdrow’). Under what circumstances would this idea be effective?

trovato	trovo
trovavano	trovata
trovava	trovare
trovati	trovarsi
trova	trovano

Problem 6 (16 points)

In conjunctive Boolean queries a query consists of a set of clauses ANDed together. Some of the individual clauses may contain ORs or NOTs. Consider the following Boolean queries for the questions below. Assume no stemming is being performed.

Query 1: (apple OR lemon) AND (pie OR tart)
Query 2: (apple OR lemon) AND (pie OR tart) AND recipe
Query 3: (apple OR lemon) AND (pie OR tart) AND recipe AND (NOT nutmeg)

(a) Which of queries 1, 2, or 3, would you expect to return the fewest number of documents? Explain why.

(b) In Chapter 1 of the course text the *intersection* algorithm is presented for efficiently creating a partial result based on two terms (or clauses, or partial results) that are being ANDed together. It assumes that docids in postings lists are in sorted (i.e., increasing) order. Describe how the basic intersection algorithm works and provide a short example.

(c) Now consider queries like Query 4 below that contain a single negation. How can the intersection algorithm be modified to efficiently resolve such queries?

Query 4: cinnamon AND (NOT nutmeg)

(d) [Queries 1-4 are not needed for this part.] Name two shortcomings of the Boolean retrieval model.

Problem 7 (3 points each)

Give brief responses to the following:

1. What is an ordered minimal perfect hash function, and why are they useful in text retrieval?
2. What effects on the size of a dictionary and an inverted file would you expect to observe when a stemming algorithm such as Porter's stemmer is used, compared to when it is not?
3. In a Boolean retrieval model, explain whether either precision or recall is expected to increase when stemming is used?
4. Why is interpolation necessary to produce precision-recall graphs?
5. Regarding IR test-collections, what is ‘pooling’?
6. Name three 'tracks' that have been explored in the TREC conferences?