

Name:Alex Gerbessiotis.....

ID Number:

Exam Number:

.....

Grade: 1: ... 2: ... 3: ... 4: ... 5: ... 6: ... 7: ... Total:

SOLVE **ALL the problems** IN THE SPACE PROVIDED

Read the Problems CAREFULLY!

THERE ARE 6 (SIX) PAGES THIS PAGE INCLUDED
Read and Sign the statement below at the end of the exam
Unsigned exams will be marked with grade 0 (zero).

STATEMENT

On my honor, I pledge that I have not violated the provision of the NJIT Student Honor Code.

Sign below at the end of the exam

Signature

.....

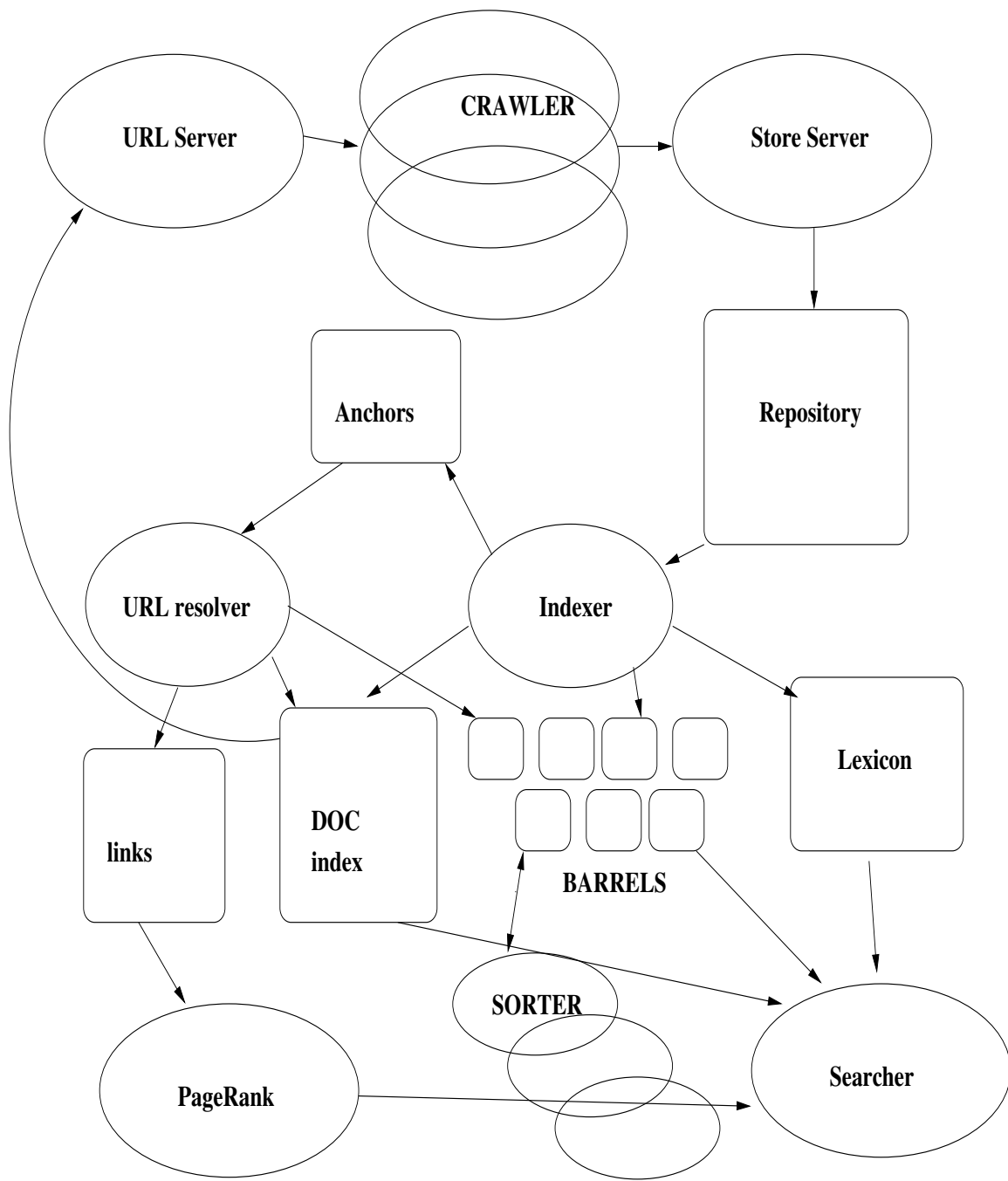


Figure 1: Google System Architecture.

Problem 1. (40 POINTS)

Give short answers to the following questions. (Use the Google System Architecture figure if you need it, for references related to Google.)

- (1) What information does the **URLserver** provide to the Crawler?
URLs
- (2) What is the functionality of the **Store Server**?
Receives web-pages from the crawler and stores them in compressed form into the Repository.
- (3) Where does Google store retrieved (by the crawler) web-pages?
Repository
- (4) Name (do not further discuss) 5 data structures used by Google.
Repository, DocIndex, Lexicon, ForwardIndex, InvertedIndex, and also Hitlists, BigFiles, Anchors, links.
- (5) What is the functionality of **links** in Google?
Stores docID pairs that show what doc points to what doc. This info is used by PageRank to compute PageRank ranks for docIDs/URLs.
- (6) What is the input for the **PageRank** computations?
links
- (7) Where does Google store the forward index?
Barrels.
- (8) Where does Google store the inverted index?
Barrels.
- (9) What do we call a frequently occurring word in a text that is not a keyword?
stopword
- (10) What do we call the operation that extracts the root of a word (and thus eliminates) its suffix/ending?
stemming

Problem 2. (20 POINTS)

Provide brief answers (one or two sentences maximum) to the following questions.

- (a) What is the difference between a full-text logical view of a document and an index-term logical view?
In full-text logical view all the words become keywords, in index-term logical view, only a fraction of the words, after possible stemming or stopwords elimination, generate index-terms, and those index-term are the keywords.
- (b) Is every word in a document a keyword, and is every keyword an index-term? Explain, or explain the differences of the expressions **word**, **keyword**, **index-term**.
Every word in a document might or might not become a keyword (for example stopwords do not become keywords).
Even if two words are to become keywords (e.g. algorithms, algorithmic) text transformations might generate one index-term for both keywords (e.g. algorithm) and the index-term is used further in indexing and query processing corresponding representing both keywords.

Turn page

Problem 3. (20 POINTS)

For a specific query involving term t , the corpus (collection of documents) includes the ten documents $d_1, \dots, d_{10}$. The first five $d_1, \dots, d_5$ contain index term t and can thus be relevant, whereas the remaining five $d_6, \dots, d_{10}$ do not contain t at all.

As a response to the query the search engine returned document

$$d_1, d_3, d_8, d_4, d_7, d_9$$

in that specific order.

(a) What is the precision of the search query? Explain.

From page 5, precision is the proportion of the retrieved documents that are relevant. Out of the 6 retrieved documents, only d_1, d_3, d_4 are relevant i.e. precision is $3/6 = 1/2$.

(b) What is the recall of the search query? Explain.

From page 5, recall is the proportion of the relevant documents that are retrieved. Out of the 5 relevant documents $d_1, \dots, d_5$, three of them d_1, d_3, d_4 have been retrieved i.e. recall is $3/5$.

Problem 4. (12 POINTS)

Which of the AND, OR, NOT are operators in (i) Google, and (ii) Bing?

(i) For GOOGLE AND, OR are operators, NOT is not but - is (but not asked).

(ii) For Bing all three are operators.

Problem 5. (8 POINTS)

Give the unary and Elias- γ codes for 5.

(i). The unary code for 5 is 111110.

(ii). The Elias- γ code for 5 is 11001

Note that 5 in binary is 101.

w_d is k_d in unary where $k_d = \lfloor \log 5 \rfloor = 2$, i.e. $w_d = 110$.

w_r is $k_r = k - 2^{k_d}$ in binary written using $k_d = 2$ bits. Since $k_r = 5 - 2^2 = 1$ this is $w_r = 01$ using $k_d = 2$ bits.

The concatenation $w_d w_r$ is the code.

(In other words from the binary code of 5 which is 101 isolate the left most one which is in position two from the rest 01. The position of the left-most one gives the unary 110 and the rest 01 and thus 11001 results.)

Problem 6. (25 POINTS)

You have a machine with 4000MBytes of free memory. We want to use all this memory to store a hash table representing the lexicon of a search engine. The average length of a word w in the lexicon is around 11 characters UNICODE. The hash table will store a pointer p that points to a location in memory where the string word w can be stored; pointer p is 4 bytes long. In addition to p a wordID needs to be locatable in addition to w . Organize the hash table so that you extract from it maximum performance. How many words w can the lexicon accommodate? Explain.

11 characters UNICODE is 22 bytes. Add 2 more bytes for NULL in UNICODE to get 24 bytes on the average for every word w . Let n be the total number of words. Writing down all the words as a long string (array in memory) would require $24n$ bytes total. Indexes to that array can be the pointer p values used. A hash table with open-addressing will accommodate optimally the n words if it has twice as many slots $2n$ so that the load-factor is kept at 50%. Each slot would accommodate a pointer p (4 bytes) and a wordID (3 or 4 bytes). We guess 4 bytes for a wordID (i.e. more than 16,000,000 words would be stored, since 3 bytes for wordID could only accommodate no more than around 16,000,000 words). (The same consideration could be used for p , except that we were told to use 4 bytes for p .) This way 8 bytes per slot of the hash table gives rise to $8 \times 2n = 16n$ for hash table size. Adding the two $16n + 24n = 40n$ and equating to 4000MBytes and solving for n we get 100,000,000 words can be accommodated by the lexicon.

The value of n determined also confirm the choice of using 4 bytes instead of 3 for wordID, since 3 bytes could only have accommodated 16,000,000 words.)

Problem 6 (continued)

```

----- ^
|          | |
|<-4 +4 --->| 2n slots
|          | |
----- v
(4+4)*2n+ (11+1)*2*n = 16n+24n=40n    --> 40n =4000MB --> n=100,000,000 (roughly)

```

algae\0algorithm\0binary\0
<----- (11+1)*2*n=24n ----->

Problem 7. (25 POINTS)

Apply the simhash algorithm on the text using b vectors with $b = 4$.

web search index web term web search term

One solution is

```

web 3 0 1 0 1
search 2 1 0 1 0
index 1 0 0 1 1
term 2 1 1 0 0

```

```

V = < 0 0 0 0 0 >
V = <-3+2-1+2
      +3-2-1+2
      -3+2+1-2
      +3-2+1-2 >
V = < 0 2 -2 0 0 >
H = <0 1 0 0 0 >

```

The hash of the document is vector H.

End of Exam 1