Øving 3 IR

Oppgave 1

1. **Given the following list of characters with frequencies, calculate the Huffman codes by using the canonical tree**

Symbol	Freq	Canonical code	Code length
i	56	111	3
e	52	110	3
b	35	101	3
f	35	011	3
l	35	010	3
d	31	001	3
a	22	1001	4
g	22	1000	4
h	12	0001	4
c	11	0000	4

What is the average length of the code?

Avg length = (freq(a)*len(a) + freq(b)*len(b) + freq(c)*len(c) + freq(d)*len(d) + freq(e)*len(e) + freq(f)*len(f) + freq(g)*len(g) + freq(h)*len(h) + freq(i)*len(i) + freq(l)*len(l)) / freqTotal

freqTotal = 311
Avg length = (88 + 105 + 44 + 93 + 156 + 105 + 88 + 48 + 168 + 105) / 311 = 3.215

What would be the average length of the code if the frequency of the letters was equal?

Avg length = (len(a) + len(b) … len(l)) / nrOfCharacters
Avg length = (3 + 3 + 3 + 3 + 3 + 3 + 4 + 4 + 4 + 4 / 10 = 3.4

Show even the canonical Huffman tree

0’s to the left and 1’s to the right. O=node

```
                      O
                 /        \
              /             \
             O                O
         /     \            /     \
       O        O          O       O
     /   \    /   \      /  \     /   \
    O    D    L      F   O    B    E   I
   /  \                /    \
  C    H               G     A
```

2. **Decode the string 1110010100100001**

   1 = B
   1 = B
   1 = B
   001 = Q
   010 = T
   010 = T
   0001 = L

Oppgave 2

1. **Give a short explanation of Lucene.**

   Lucene is a library designed for information retrieval. It is generally used for text indexing and search. It is based on the theory that all documents contains text fields, and every file where you can extract text fields can be seen that way. That way Lucene can index documents of several formats.

2. **Copy the console output after doing the indexing**

```
Indexing to directory 'C:\Users\juliejk\Documents\Github\assignment3\index'...
adding C:\Users\juliejk\Documents\Github\assignment3\apple\doc1.txt
adding C:\Users\juliejk\Documents\Github\assignment3\apple\doc10.txt
adding C:\Users\juliejk\Documents\Github\assignment3\apple\doc2.txt
adding C:\Users\juliejk\Documents\Github\assignment3\apple\doc3.txt
adding C:\Users\juliejk\Documents\Github\assignment3\apple\doc4.txt
adding C:\Users\juliejk\Documents\Github\assignment3\apple\doc5.txt
adding C:\Users\juliejk\Documents\Github\assignment3\apple\doc6.txt
adding C:\Users\juliejk\Documents\Github\assignment3\apple\doc7.txt
adding C:\Users\juliejk\Documents\Github\assignment3\apple\doc8.txt
adding C:\Users\juliejk\Documents\Github\assignment3\apple\doc9.txt
563 total milliseconds
```

Explain the steps involved in the indexing process according to the source code

The main method creates an indexpath and a docPath from the arguments. Next it checks if the docPath is not equal or that the directory was not found. if all checks out, it starts to initialize the analyser and the indexWriterConfig. Then the indexWriterConfig sets a open mode to create new indexes to the directory. Then it initialize the indexWriter and runs the method indexDocs with the indexWriter and docDir. When indexDocs is done running it closes the indexWriter. It also keep track of the time it takes to index the documents in the directory.

The indexDocs method takes in a indexWriter and a file and starts to check if the file can be read, and if so it check whether the file is a directory, and if it is it puts the files in a list. If the file in the files list is not empty it runs the indexDocs with the file, and keeps doing that for each file in the directory. So if the file isn’t a directory it initialize the FileInputStream and tries to run it on the file. Then is tries to create a new document, then create a Field, and add the Field to the document. Then the document adds the last modified value and the contents of the file. Last it check if it needs to update or add the document, and then adds or updates the document. And finally it closes the FileInputStream.

3. Copy the result from the console for each query

- **Grape**

```
Enter query:
grape
Searching for: grape
5 total matching documents
1. C:\Users\juliejk\Documents\Github\assignment3\apple\doc9.txt
2. C:\Users\juliejk\Documents\Github\assignment3\apple\doc5.txt
3. C:\Users\juliejk\Documents\Github\assignment3\apple\doc2.txt
4. C:\Users\juliejk\Documents\Github\assignment3\apple\doc3.txt
5. C:\Users\juliejk\Documents\Github\assignment3\apple\doc10.txt
Press (q)uit or enter number to jump to a page.
```

- **Melon Grape**

```
Enter query:
melon grape
Searching for: melon grape
9 total matching documents
1. C:\Users\juliejk\Documents\Github\assignment3\apple\doc9.txt
2. C:\Users\juliejk\Documents\Github\assignment3\apple\doc5.txt
3. C:\Users\juliejk\Documents\Github\assignment3\apple\doc10.txt
4. C:\Users\juliejk\Documents\Github\assignment3\apple\doc3.txt
5. C:\Users\juliejk\Documents\Github\assignment3\apple\doc6.txt
6. C:\Users\juliejk\Documents\Github\assignment3\apple\doc2.txt
7. C:\Users\juliejk\Documents\Github\assignment3\apple\doc8.txt
8. C:\Users\juliejk\Documents\Github\assignment3\apple\doc1.txt
9. C:\Users\juliejk\Documents\Github\assignment3\apple\doc4.txt
Press (q)uit or enter number to jump to a page.
```

- **Apple Melon Grape**

```
Enter query:
apple grape melon
Searching for: apple grape melon
10 total matching documents
1. C:\Users\juliejk\Documents\Github\assignment3\apple\doc10.txt
2. C:\Users\juliejk\Documents\Github\assignment3\apple\doc3.txt
3. C:\Users\juliejk\Documents\Github\assignment3\apple\doc5.txt
4. C:\Users\juliejk\Documents\Github\assignment3\apple\doc2.txt
5. C:\Users\juliejk\Documents\Github\assignment3\apple\doc9.txt
6. C:\Users\juliejk\Documents\Github\assignment3\apple\doc1.txt
```

```
7. C:\Users\juliejk\Documents\Github\assignment3\apple\doc4.txt
8. C:\Users\juliejk\Documents\Github\assignment3\apple\doc7.txt
9. C:\Users\juliejk\Documents\Github\assignment3\apple\doc6.txt
10. C:\Users\juliejk\Documents\Github\assignment3\apple\doc8.txt
Press (q)uit or enter number to jump to a page.
```

What query model is used here? Do returned documents contain all query terms together (AND) or are all documents containing any of them returned (OR)?

When we ran the “melon grape” search we got the doc1 out which only contains melon and apple, not grape, therefore we concluded that the query model used here are OR, otherwise the doc1 wouldn’t be printed out. The same goes for doc2, doc4, doc6, doc8 which all have one but not both.