Laporan Pemerolehan Informasi UTS 2

Pencarian menggunakan Inverted Index

Dosen Pengampu : JB. Budi Darmawan S.T., M.Sc.

Disusun oleh

Nama : Dyline Melynea Fernandez

NIM : 185314125

PROGRAM STUDI INFORMATIKA

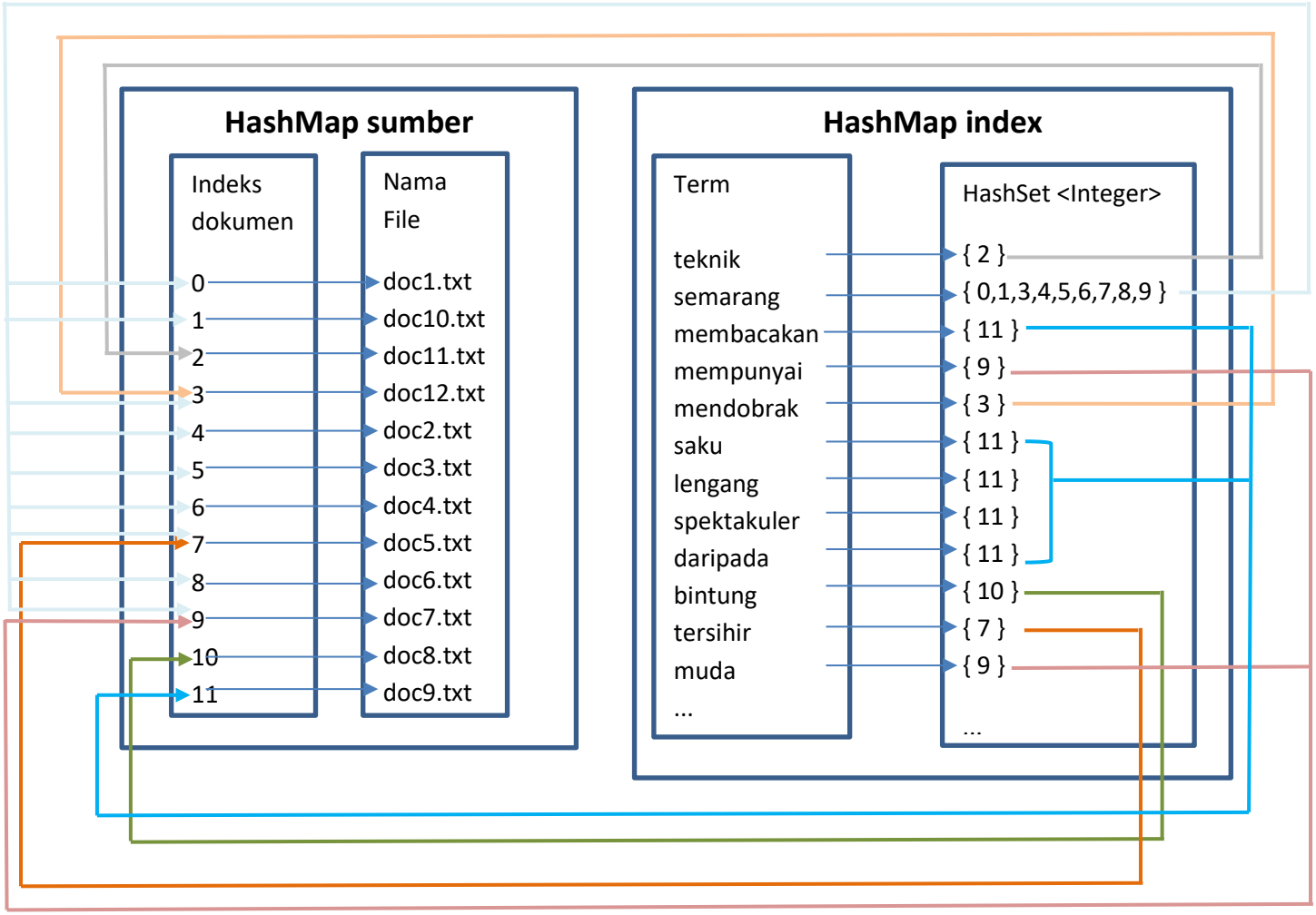
FAKULTAS SAINS & TEKNOLOGI

UNIVERSITAS SANATA DHARMA

A. Rancangan struktur data

Untuk **index** dan **sumber** menggunakan HashMap yang prinsipnya seperti kamus/dictionary. Namun value dari HashMap **index** dibuat struktur datanya yaitu HashSet yang merupakan himpunan. Sehingga value dari key HashMap **index** bisa lebih dari 1 elemen. **index** merupakan pemetaan antara term sebagai key dan id dokumen/indeks dokumen sebagai valuenya, **sumber** merupakan pemetaan antara id dokumen/indeks dokumen sebagai key dan nama file dokumen sebagai valuenya.

Ilustrasi:



Konektor warna-warni hanyalah gambaran di method search, setelah didapatkan himpunan integer, maka memanggil HashMap **sumber** untuk mendapatkan nama file dokumen.

Gambaran output HashMap **sumber** dan **index** di program

```
[0=doc1.txt, 1=doc10.txt, 2=doc11.txt, 3=doc12.txt, 4=doc2.txt, 5=doc3.txt, 6=doc4.txt, 7=doc5.txt, 8=doc6.txt, 9=doc7.txt, 10=doc8.txt, 11=d
[teknik=[2], semarang=[0, 1, 3, 4, 5, 6, 7, 8, 9], membacakan=[11], mempunyai=[9], mendobrak=[3], saku=[11], lengang=[11], spektakuler=[11],
```

B. Algoritma searching more than 1 keyword

1. Pisahkan setiap term/kata keywords (patokan pisah dengan whistespace)
2. Buat objek HashSet untuk menampung index term (tipe Integer)
3. Cari irisan index dari semua keywords (gunakan fungsi retainAll dari kelas HashSet)
4. Apabila ukuran dari objek hashset yang menampung index term adalah 0, maka cetak "tidak ditemukan"
5. Lakukan perulangan sebanyak jumlah elemen objek HashSet
6. Cetak mapping values dari key yang bersesuaian iterasi foreach objek HashSet

Penjelasan:

Keyword dipisah terlebih dahulu untuk setiap katanya, kemudian membuat objek HashSet untuk menampung indeks term. Untuk awal pertama membuat objek, isi parameter konstruktor HashSet diisi dengan **index.get(keywords[0].toLowerCase())** untuk inisialisasi. Setelahnya isi dari objek HashSet tersebut (**res**), akan mengalami perulangan sebanyak banyak elemen keyword yang sudah dipisahkan per kata. Pada perulangan tersebut, akan ditetapkan value **res** yang beririsan dengan hasil indeks term.

Kemudian selanjutnya memfilter kondisi jika ukuran objek **res** adalah 0, yang artinya tidak ada index term tersebut, maka cetak "tidak ditemukan".

Setelah sudah melakukan kondisi di atas, di luar kondisi tersebut dilakukan foreach loop sebanyak elemen **res**. Dan mencetak atribut **sumber** tipe HashMap yang menampung indeks term sebagai key dengan nama filenya sebagai value. Hasil atribut itu adalah dokumen yang mewadahi keyword yang dicari.

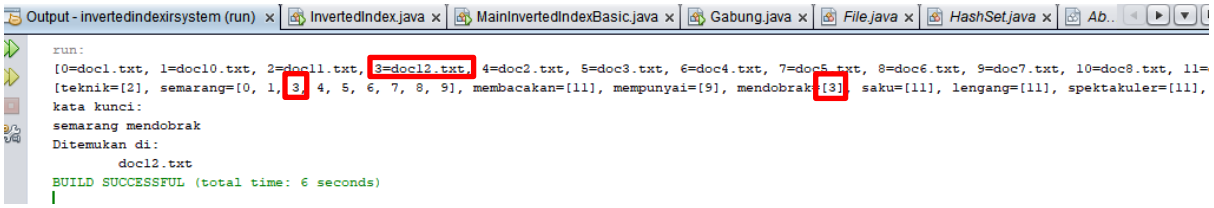
C. Program (captured)

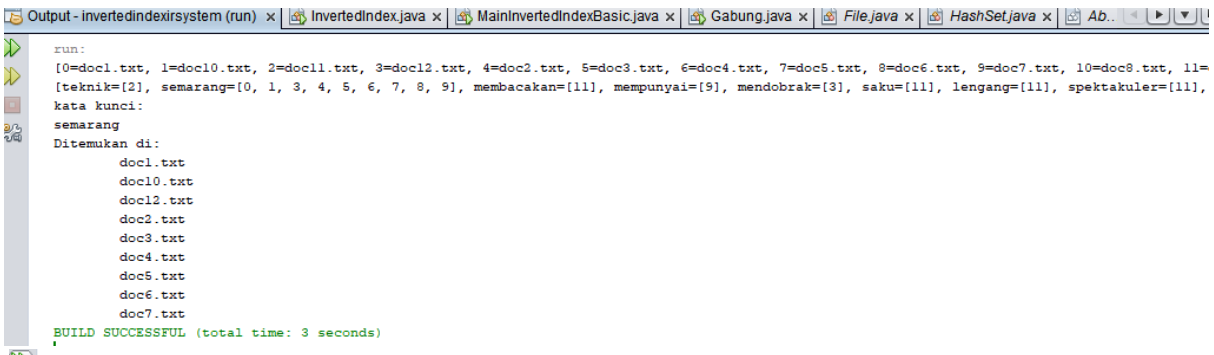
```
1   // @author dyline
2
3   import java.io.*;
4   import java.util.*;
5
6   class Index {
7
8       Map<Integer, String> sumber;
9       HashMap<String, HashSet<Integer>> index;
10
11      Index() {
            sumber = new HashMap<Integer, String>();
            index = new HashMap<String, HashSet<Integer>>();
14      }
15
16      public void buatIndex(String[] files) {
17          int i = 0;
18          for (String namaFile : files) {
19
20              try (BufferedReader file = new BufferedReader(new FileReader("..\\invertedindexirsystem\\Koleksi\\"
21                      + namaFile))) {
22                  BufferedReader stopwords = new BufferedReader(
                            new FileReader("..\\invertedindexirsystem\\stopwordsin.txt"));
24                  sumber.put(i, namaFile);
25                  String ln;
26                  String readStopwords;
27                  String cleanTerms = "";
28                  while ((ln = file.readLine()) != null) {
29                      readStopwords = stopwords.readLine();
30                      String[] terms = ln.split("\\W+");
31                      String[] stopword = readStopwords.split("\\W+");
32
33                      for (String term : terms) {
34                          term = term.toLowerCase();
                            if (!term.equals(stopword)) {
```

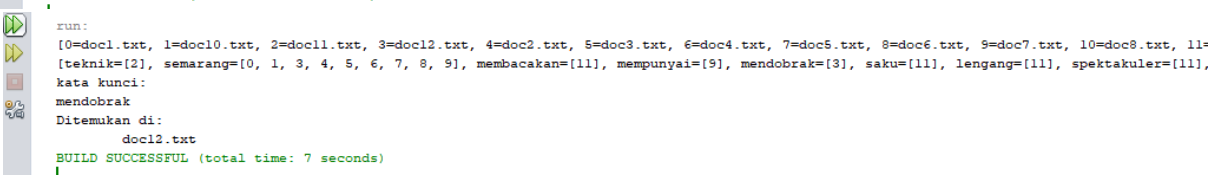
```
                    cleanTerms = term;
                }
                if (!index.containsKey(cleanTerms)) {
                    index.put(cleanTerms, new HashSet<Integer>());
                }
                index.get(cleanTerms).add(i);
            }
        }
    } catch (IOException e) {
        System.out.println("File " + namaFile + " tidak ditemukan. skip");
    }
    i++;
}
index.remove("");
System.out.println(sumber.entrySet());
System.out.println(index.entrySet());
}

public void search(String key) {
    String[] keywords = key.split("\\W+");
    HashSet<Integer> res = new HashSet<Integer>(index.get(keywords[0].toLowerCase()));
    for (String kw : keywords) {
        res.retainAll(index.get(kw));
    }

    if (res.size() == 0) {
        System.out.println("Tidak ditemukan");
        return;
    }
    System.out.println("Ditemukan di: ");
    for (int num : res) {
        System.out.println("\t" + sumber.get(num));
    }
}
}
```

```
public class InvertedIndex {

    public static void main(String args[]) throws IOException {
        Index index = new Index();
        String path = "..\\invertedindexirsystem\\Koleksi";
        File name = new File(path);
        if (name.exists()) {
            if (name.isDirectory()) {
                String directory[] = name.list();
                index.buatIndex(directory);
                System.out.println("kata kunci: ");
                BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
                String keywords = in.readLine();

                index.search(keywords);
            }
        }
    }
}
```

Output:


Output - invertedindexirsystem (run)
InvertedIndex.java
MainInvertedIndexBasic.java
Gabung.java
File.java
HashSet.java
run:
[0=doc1.txt, 1=doc10.txt, 2=doc11.txt, 3=doc12.txt, 4=doc2.txt, 5=doc3.txt, 6=doc4.txt, 7=doc5.txt, 8=doc6.txt, 9=doc7.txt, 10=doc8.txt, 11=
[teknik=[2], semarang=[0, 1, 3, 4, 5, 6, 7, 8, 9], membacakan=[11], mempunyai=[9], mendobrak=[3], saku=[11], lengang=[11], spektakuler=[11],
kata kunci:
semarang mendobrak
Ditemukan di:
doc12.txt
BUILD SUCCESSFUL (total time: 6 seconds)


Output - invertedindexirsystem (run)
InvertedIndex.java
MainInvertedIndexBasic.java
Gabung.java
File.java
HashSet.java
run:
[0=doc1.txt, 1=doc10.txt, 2=doc11.txt, 3=doc12.txt, 4=doc2.txt, 5=doc3.txt, 6=doc4.txt, 7=doc5.txt, 8=doc6.txt, 9=doc7.txt, 10=doc8.txt, 11=
[teknik=[2], semarang=[0, 1, 3, 4, 5, 6, 7, 8, 9], membacakan=[11], mempunyai=[9], mendobrak=[3], saku=[11], lengang=[11], spektakuler=[11],
kata kunci:
semarang
Ditemukan di:
doc1.txt
doc10.txt
doc12.txt
doc2.txt
doc3.txt
doc4.txt
doc5.txt
doc6.txt
doc7.txt
BUILD SUCCESSFUL (total time: 3 seconds)


run:
[0=doc1.txt, 1=doc10.txt, 2=doc11.txt, 3=doc12.txt, 4=doc2.txt, 5=doc3.txt, 6=doc4.txt, 7=doc5.txt, 8=doc6.txt, 9=doc7.txt, 10=doc8.txt, 11=
[teknik=[2], semarang=[0, 1, 3, 4, 5, 6, 7, 8, 9], membacakan=[11], mempunyai=[9], mendobrak=[3], saku=[11], lengang=[11], spektakuler=[11],
kata kunci:
mendobrak
Ditemukan di:
doc12.txt
BUILD SUCCESSFUL (total time: 7 seconds)