Telekomunikacja - laboratorium	Studia stacjonarne - inzynierskie

Nazwa zadania	Ćwiczenie nr 3. Implementacja algorytmu kodowania Huffmana

Dzień	Poniedziałek	Godzina	8:15	Rok akademicki	2019/20

Imię i Nazwisko	Patryk Kolanek
Imię i Nazwisko	Kacper Świercz

Opis programu, rozwiązania problemu.

Ćwiczenia polega na zaimplementowaniu algorytmu statycznego kodowania Huffmana w programie, który przesyła plik za pomocą gniazd sieciowych na inny komputer. Algorytm ten jest wykorzystywany często jako ostatni etap w różnych systemach kopresji danych. Takiej jak MP3 lub JPEG.

Gniazdo sieciowe jest to para, adres IP oraz numer portu. Numer portu działa w warstwie transportowej modelu sieciowego TCP/IP. Port identyfikuje proces, który wymienia dane przez gniazdo. Dzięki numerowi portu można zidentyfikować do którego procesu mają trafić pakiety.

Kodowanie Huffmana służy do bezstratnej kompresji danych. Algorytm ten nie jest efektywny obliczeniowo.
Zadanie algorytmu jest utworzenie słownika słów kodowych, który zawiera słowa kodowe dla danych znaków (bajtów). Im częściej dany znak występuje, tym krótsze jest jego słowo kodowe.
Do tego celu na początku liczone jest prawdopodobieństwo (czyli ilość wystąpienia) danego znaku. Posiadając prawdopodobieństwo wystąpienia danego znaku, układane jest rosnąco drzewo binarne, które zawiera dany znak oraz jego prawdopodobieństwo. Dopóki drzewo posiada więcej niż jeden korzeń, dwa korzenie o najmniejszych prawdopodobieństwach są ze sobą sumowane i wstawiane do drzewa przypisując lewej gałęzi 0, a prawej 1. Długość słowa kodowanego jest równa głębokości znaku w drzewie, a jego wartość zależy od położenia w drzewie.

Kodowanie pliku polega na zastąpieniu danych znaków, słowami kodowymi utworzonymi za pomocą algorytmu Huffmana. Do odkodowania takiego pliku potrzebny jest słownik kodowy, który jest przesyłany razem z plikiem. Proces dekompresji polega na zamianie zakodowanych znaków na konkretne bajty.

Najważniejsze elementy kodu programu z opisem.

Gniazdo jest punktem końcowym dwukierunkowej komunikacji pomiędzy procesami. Do realizacji zadania postanowiliśmy skorzystać architektury klient-serwer. Serwer, program który nasłuchuje na danym porcie. Do realizacji tego zadania wykorzystana została klasa ServerSocket.

```
ServerSocket serverSocket = new ServerSocket(port);
```

Metoda accept() oczekuje na zgłoszenie klienta, który chce nadawać na dany port. Metoda ta zwraca obiekt klasy Socket, który reprezentuje gniazdo.

```
Socket socket = serverSocket.accept();
```

Po stronie klienta do próby połączenia gniazda służy konstruktor klasy Socket.

```
Socket socket = new Socket(address, port);
```

Podczas wymiany danych mamy do czynienia ze standardowymi strumieniami w języku programowania Java, które można pobrać metodami getOutputStream i getInputStream.

```
socket.getOutputStream();
socket.getInputStream();
```

Obiekty przesyłane przez strumienie do serwera muszą podlegać serializacji przez wirtualną maszynę Javy. Dlatego obiekty klas, które są przesyłane przez strumień implementują interfejs Serializable.

```
* implements Serializable { }
```

Do implementacji algorytmu Huffmana potrzebna jest struktura, która zapamiętuje wartość oraz prawdopodobieństwo znaku. Każdy węzeł posiada odniesienie do węzła po lewo i po prawo.

```
public class Node implements Comparable<Node> {
     private Node leftNode;
     private Node rightNode;
     private int probability;
     private byte value;
}
```

Tworzenie drzewa binarnego polega na ułożeniu posortowanych rosnąco węzłów w drzewo binarne, gdzie dwa węzły o dwóch najmniejszych prawdopodobieństwach.

```
while (rootNodes.size() > 1) {
     Collections.sort(rootNodes);
     Node firstNode = rootNodes.get(0);
     rootNodes.remove(0);
     Node secondNode = rootNodes.get(0);
     rootNodes.remove(0);
     Node root = new Node((firstNode.getProbability() + secondNode.getProbality(), (byte) 0);
     root.setLeftNode(firstNode);
     root.setRightNode(secondNode);
     rootNodes.add(root);
}
```

Do utworzenia słownika słów kodowych wykorzystano rekurencyjny algorytm przeszukiwania drzewa binarnego.

```
private void createDictionary(Node node, String key) {
        if (node.getLeftNode() == null) {
            if (node.getRightNode() == null) {
                dictionary.add(new Element(node.getValue(), key));
                return;
            }
            createDictionary(node.getRightNode(), key + "1");

        }
        if (node.getRightNode() == null) {
            createDictionary(node.getLeftNode(), key + "0");
        }
        createDictionary(node.getLeftNode(), key + "0");
        createDictionary(node.getRightNode(), key + "1");
}
```

Wykorzystano standardowe strumienie dla plików FileInputStream oraz FileOutputStream.

Podsumowanie wnioski.

Algorytm kompresji Huffmana jest algorytmem służącym do kompresji danych. Jest to algorytm łatwy w implementacji, niestety jego złożoność obliczeniowa nie jest efektywna. Kolejnym problemem jest konieczność tworzenia słów kodowych za każdym razem od nowa. Dodatkowo do przesyłanego pliku, trzeba przesłać słownik słów kodowych.