МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г.
ШУХОВА» (БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №5
по дисциплине: Теория информации
тема: «»

Выполнил: ст. группы ПВ-223
Дмитриев Андрей Александрович

Проверил:
Твердохлеб Виталий Викторович

Белгород 2024 г.

Содержание:

1. Тема лабораторной работы.
2. Цель лабораторной работы.
3. Текст задания к работе.
4. Выполнение задания.
5. Вывод по работе.

Цель лабораторной работы: Построить обработчик, реализующий функцию арифметического кодирования.
В качестве исходных данных, подлежащих обработке, использовать последовательности из работы №2.
Для полученных результатов рассчитать показатели сжатия. Сравнить с полученными в работе №2.

Ход работы:

```
class ArithmeticEncoder {
    private ArrayList<Map.Entry<Character, Double>> tablePrevProbabilities;
    private MathContext MATH_CONTEXT = new MathContext(5);

    private static HashMap<Character, Integer> getAmountsOfSymbol(String
source) {
        HashMap<Character, Integer> amountsOfSymbol = new HashMap<>();
        for (var symbol : source.toCharArray()) {
            amountsOfSymbol.compute(symbol, (k, v) -> v != null ? v + 1 :
1);
        }

        return amountsOfSymbol;
    }

    private void setProbabilityLine(String source) {
        // подсчёт количества символов
        HashMap<Character, Integer> amountsOfSymbol =
getAmountsOfSymbol(source);
        // подсчёт частот символов
        HashMap<Character, Double> probabilitiesOfSymbol = new HashMap<>();
        for (var pairSymbolNCounter : amountsOfSymbol.entrySet()) {
            probabilitiesOfSymbol.put(pairSymbolNCounter.getKey(), (double)
pairSymbolNCounter.getValue() / source.length());
        }
        // сортировка по частоте
        tablePrevProbabilities = new ArrayList<>();
        for (var pairSymbolNProbability : probabilitiesOfSymbol.entrySet())
{
            int i = 0;
            while (i < tablePrevProbabilities.size() &&
                    tablePrevProbabilities.get(i).getValue() >
pairSymbolNProbability.getValue())
                i++;

            tablePrevProbabilities.add(i, pairSymbolNProbability);
        }
        // преобразование в массив окончаний промежутков
        for (int i = 1; i < tablePrevProbabilities.size(); i++) {
            var prevProbability = tablePrevProbabilities.get(i -
1).getValue();
            var currProbability = tablePrevProbabilities.get(i).getValue();
            tablePrevProbabilities.get(i).setValue(prevProbability +
currProbability);
        }
    }

    private BigDecimal getHighRange(char symbol) {
        for (var pairSymbolNProbability : tablePrevProbabilities)
            if (pairSymbolNProbability.getKey().equals(symbol))
                return new BigDecimal(pairSymbolNProbability.getValue(),
MATH_CONTEXT);

        throw new NullPointerException();
    }

    private BigDecimal getLowRange(char symbol) {
        if (tablePrevProbabilities.get(0).getKey().equals(symbol))
            return new BigDecimal("0.", MATH_CONTEXT);

        for (int i = 1; i < tablePrevProbabilities.size(); i++)
            if (tablePrevProbabilities.get(i).getKey().equals(symbol))
                return new BigDecimal(tablePrevProbabilities.get(i -
1).getValue(), MATH_CONTEXT);
```

```
        throw new NullPointerException();
    }

    /**
     * Кодирование текста арифметическим кодированием
     *
     * @param text - исходный текст
     * @return - код текста
     * @throws NullPointerException
     */
    public BigDecimal encode(String text) throws NullPointerException {
        setProbabilityLine(text);

        try {
            BigDecimal low = getLowRange(text.charAt(0));
            BigDecimal high = getHighRange(text.charAt(0));
            for (int i = 1; i < text.length(); i++) {
                BigDecimal tempLow =
high.subtract(low).multiply(getLowRange(text.charAt(i)));
                high =
low.add(high.subtract(low).multiply(getHighRange(text.charAt(i))));

                low = low.add(tempLow);
            }

            return (low.add(high)).divide(new BigDecimal("2."));

        } catch (NullPointerException e) {
            e.fillInStackTrace();
            throw new NullPointerException();
        }
    }

    /**
     * Раскодирование текста арифметическим кодированием
     *
     * @param encodedValue - код текста
     * @return строку исходного текста
     * @throws NullPointerException
     */
    public String decode(BigDecimal encodedValue) throws
NullPointerException {
        var sb = new StringBuilder();
        var encodedValueCopy = new BigDecimal(encodedValue.toString());
        while
(encodedValueCopy.subtract(BigDecimal.valueOf(0.5)).abs().compareTo(BigDecim
al.valueOf(0.000001)) > 0) {
            for (var pairSymbolNProbability : tablePrevProbabilities) {
                if
(encodedValueCopy.compareTo(BigDecimal.valueOf(pairSymbolNProbability.getVal
ue())) > 0) {
                    continue;
                }

                sb.append(pairSymbolNProbability.getKey());
                try {
                    encodedValueCopy =
(encodedValueCopy.subtract(getLowRange(pairSymbolNProbability.getKey())))

.divide(getHighRange(pairSymbolNProbability.getKey())

.subtract(getLowRange(pairSymbolNProbability.getKey())));
                } catch (NullPointerException | ArithmeticException e) {
                    e.fillInStackTrace();
                    throw e;
                }
```

```java
                break;
            }
        }

        return sb.toString();
    }
}

public class Main {
    public static void main(String[] args) {
        var encoder = new ArithmeticEncoder();

        var encodedValue = encoder.encode("sssehhhh");
        System.out.println(encodedValue);
        System.out.println(encoder.decode(encodedValue));

        encodedValue = encoder.encode("ЭТОТ_МЕТОД_ЛУЧШЕ_ХАФФМАНА");
        System.out.println(encodedValue);
        System.out.println(encoder.decode(encodedValue));

        encodedValue = encoder.encode("victoria nulla est, quam quae
confessos animo quoque subjugat hostes");
        System.out.println(encodedValue);
        System.out.println(encoder.decode(encodedValue));

        encodedValue = encoder.encode("""
                Не жалею, не зову, не плачу,
                Все пройдет, как с белых яблонь дым.
                Увяданья золотом охваченный,
                Я не буду больше молодым.

                Ты теперь не так уж будешь биться,
                Сердце, тронутое холодком,
                И страна березового ситца
                Не заманит шляться босиком.

                Дух бродяжий! ты все реже, реже
                Расшевеливаешь пламень уст
                О моя утраченная свежесть,
                Буйство глаз и половодье чувств.

                Я теперь скупее стал в желаньях,
                Жизнь моя? иль ты приснилась мне?
                Словно я весенней гулкой ранью
                Проскакал на розовом коне.

                Все мы, все мы в этом мире тленны,
                Тихо льется с кленов листьев медь…
                Будь же ты вовек благословенно,
                Что пришло процвесть и умереть.
                """);
        System.out.println(encodedValue);
        System.out.println(encoder.decode(encodedValue));
    }
}
```

Пример работы программы:

0.81142283333708990551414544889863896335260139388928000000000000000000000000000000000000000000000000000000000000000000000000000000

victoria nulla est, quam quae confessos animo quoque subjugat hostes

0.97300263173919095175143377233754905493676973234097593674167998042564676558584819986276284979265845329756270416747368293074055523

183996816993003683881475280974896497157536112110460005866834730523682185071950654305787717059334179963638291333757042551019410145377413293279227601966611407956100667844251344935383104566720244573658816849617333195593488470875245209817925847342363807723745732169177496717968110915786533993472363850637757902828087383755249699128184737634155638905468274209708074193385055202624097695591537814821051203185910357015779456703723765839730813485547591010623329488599967220262792667811221237423708803117369004095516322412945816746303577137790775677859294889423311408945647306842800844449931952792896919288044550982208883645479295033366720766413906902818537541873101260335107422961797422597749623535175026458248751454243645369428535594077350836470069579616465069446228375588357423965602128824874942985836863216120625851704583357867235669201997611918689877930486769335738546284194478279941836100579303802227142156596493022449565363846873148452022095871863966515978107150791180923595248876744321086260963145877021939778331061599282169739040328842924542001678427858180305179684590364297758608602321902507143038361185992316528782711151782097373084525694492599610047898006285354516066330811234377450360940140936292922795712751672756276527004123417541723754385173277221864192822181918060389645582904233089929924321843933111539136788092601687036307935593779801592566193482740300650165180090156704236444591381757071205638957466234253534317848450895166337234538982940652337139956702582560055394060400221970021835595055663127479600001255578626837899548360635352691964501885834496667735897119439539474578125065092293112140932105864061994152902490756553541367878473126061039633761341541629455045359773676904796514372328506650031990848555329622668330403581039640503183410519643725211518280264086936708863627036418100788733848991031632109641457798185717843868519911013412639197935971063549089737702694988265779464127315817343110131656178571444847859175561388224102304613424986781765576065303731043447474006493518558929961905954014036832462609903089054764918687257742667018291942211922700710328873235233689205960716806895987313446748780081104789004019164677621450377836636318622370834553270762716996538014194866261114373467507981888331176992587079402356992082309052499854687756164879145232889145782264810236726582643871006866916844455559501026218298018342430507329451110372915688969029353935084154462637924851622018335634254413606718741733766118649779448437557971557334107425484278088408655799559624751808039423476272739348818750956235399400268639801546023115293000199283208607611826323056398339684368052817114657364172333462995094569301729046288100191386844012443957307707528034082210163254367486399555612810928528854516438824438339956458426511085123474482136929664935057149284334082921994910887214430112532831223642717482543890606778511628

5218136104705720320000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000

Не жалею, не зову, не плачу,
Все пройдет, как с белых яблонь дым.
Увяданья золотом охваченный,
Я не буду больше молодым.

Ты теперь не так уж будешь биться,
Сердце, тронутое холодком,
И страна березового ситца
Не заманит шляться босиком.

Дух бродяжий! ты все реже, реже
Расшевеливаешь пламень уст
О моя утраченная свежесть,
Буйство глаз и половодье чувств.

Я теперь скупее стал в желаньях,
Жизнь моя? иль ты приснилась мне?
Словно я весенней гулкой ранью
Проскакал на розовом коне.

Все мы, все мы в этом мире тленны,
Тихо льется с кленов листьев медь…
Будь же ты вовек благословенно,
Что пришло процвесть и умереть.

Сравнение: Рассчитать программно не вышло из-за особенности динамической мантиссы BigDecimal в Java.

Вывод: в ходе работы изучены построить обработчик, реализующий функцию арифметического кодирования.