Họ tên: Nguyễn Việt Hoàng **Tổng bài: 29/29**

Mã sinh viên: 231230791

Mục lục

Bài tập 1: Ứng dụng cấu trúc dữ liệu Bảng chữ cái (Alphabet)

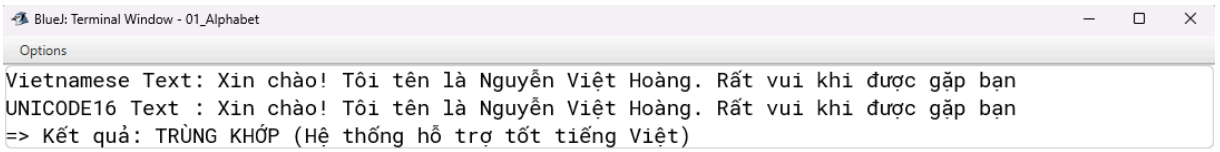
Code:

Alphabet.java

```
import java.util.Arrays;

public class Alphabet {
    public static final Alphabet BINARY = new Alphabet("01");
    public static final Alphabet OCTAL = new Alphabet("01234567");
    public static final Alphabet DECIMAL = new Alphabet("0123456789");
    public static final Alphabet HEXADECIMAL = new Alphabet("0123456789ABCDEF");
    public static final Alphabet DNA = new Alphabet("ACGT");
    public static final Alphabet LOWERCASE = new
Alphabet("abcdefghijklmnopqrstuvwxyz");
    public static final Alphabet UPPERCASE = new
Alphabet("ABCDEFGHIJKLMNOPQRSTUVWXYZ");
    public static final Alphabet PROTEIN = new Alphabet("ACDEFGHIKLMNPQRSTVWY");
    public static final Alphabet BASE64 = new
Alphabet("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/");
    public static final Alphabet ASCII = new Alphabet(128);
    public static final Alphabet EXTENDED_ASCII = new Alphabet(256);
    public static final Alphabet UNICODE16 = new Alphabet(65536);

    private char[] alphabet;
    private int[] inverse;
    private final int R;
    public Alphabet(String alpha) {
        alphabet = alpha.toCharArray();
        R = alpha.length();
        inverse = new int[Character.MAX_VALUE];
        Arrays.fill(inverse, -1);
        for (int c = 0; c < R; c++)
            inverse[alphabet[c]] = c;
    }

    public Alphabet(int radix) {
        this.R = radix;
        alphabet = new char[R];
        inverse = new int[R];
        for (int i = 0; i < R; i++) {
            alphabet[i] = (char) i;
            inverse[i] = i;
        }
    }
    public boolean contains(char c) {
        return inverse[c] != -1;
    }
    public int radix() {
        return R;
    }
    public int lgR() {
        int lgR = 0;
        for (int t = R - 1; t >= 1; t /= 2)
            lgR++;
```

```
        return lgR;
    }
    public int toIndex(char c) {
        if (c >= inverse.length || inverse[c] == -1) {
            throw new IllegalArgumentException("Character " + c + " not in
alphabet");
        }
        return inverse[c];
    }
    public char toChar(int index) {
        if (index < 0 || index >= R) {
            throw new IllegalArgumentException("index must be between 0 and " +
R);
        }
        return alphabet[index];
    }
    public int[] toIndices(String s) {
        char[] source = s.toCharArray();
        int[] target = new int[s.length()];
        for (int i = 0; i < source.length; i++)
            target[i] = toIndex(source[i]);
        return target;
    }
    public String toChars(int[] indices) {
        StringBuilder s = new StringBuilder(indices.length);
        for (int i = 0; i < indices.length; i++)
            s.append(toChar(indices[i]));
        return s.toString();
    }
}
```

Kết quả:




```
Can only enter input while your program is running
```

Bài tập 2: Sắp xếp danh sách Mã số sinh viên bằng thuật toán LSD Radix Sort

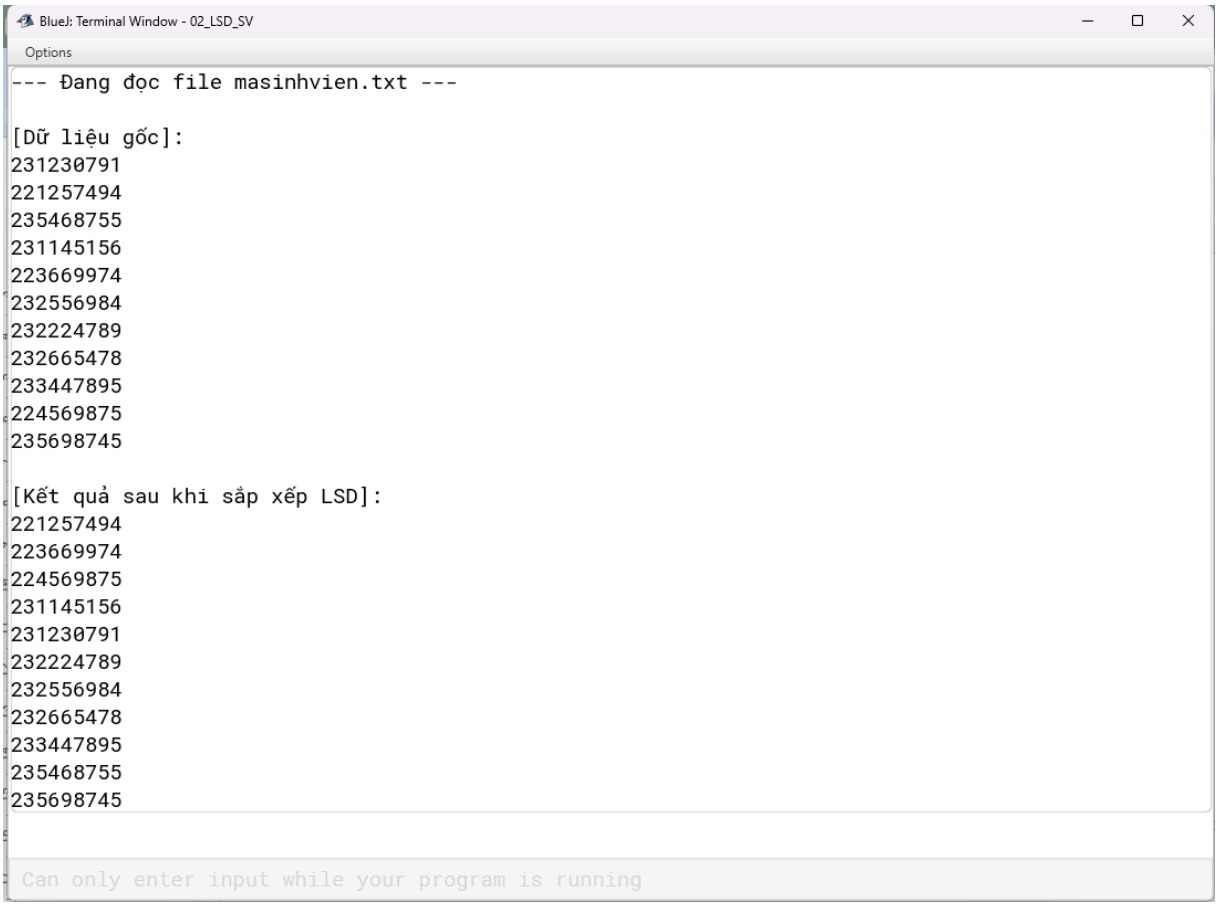
Code:

LSD.java

```
public class LSD {
    private static final int BITS_PER_BYTE = 8;
    private LSD() { }
    public static void sort(int[] a) {
        final int BITS = 32;
        final int R = 1 << BITS_PER_BYTE;
        final int MASK = R - 1;
        final int w = BITS / BITS_PER_BYTE;
        int n = a.length;
        int[] aux = new int[n];
        for (int d = 0; d < w; d++) {
            int[] count = new int[R+1];
            for (int i = 0; i < n; i++) {
                int c = (a[i] >> BITS_PER_BYTE*d) & MASK;
                count[c + 1]++;
            }
            for (int r = 0; r < R; r++)
                count[r+1] += count[r];
            if (d == w-1) {
                int shift1 = count[R] - count[R/2];
                int shift2 = count[R/2];
                for (int r = 0; r < R/2; r++)
                    count[r] += shift1;
                for (int r = R/2; r < R; r++)
                    count[r] -= shift2;
            }
            for (int i = 0; i < n; i++) {
                int c = (a[i] >> BITS_PER_BYTE*d) & MASK;
                aux[count[c]++] = a[i];
            }
            for (int i = 0; i < n; i++)
                a[i] = aux[i];
        }
    }
    public static void sort(String[] a, int w) {
        int n = a.length;
        int R = 256;   // extended ASCII
        String[] aux = new String[n];

        for (int d = w-1; d >= 0; d--) {
            int[] count = new int[R+1];
            for (int i = 0; i < n; i++)
                count[a[i].charAt(d) + 1]++;

            for (int r = 0; r < R; r++)
                count[r+1] += count[r];

            for (int i = 0; i < n; i++)
                aux[count[a[i].charAt(d)]++] = a[i];
```

```
            for (int i = 0; i < n; i++)
                a[i] = aux[i];
        }
    }
}
```

Kết quả:



Bài tập 3: Sắp xếp danh sách Họ tên sinh viên bằng thuật toán MSD Radix Sort

Code:

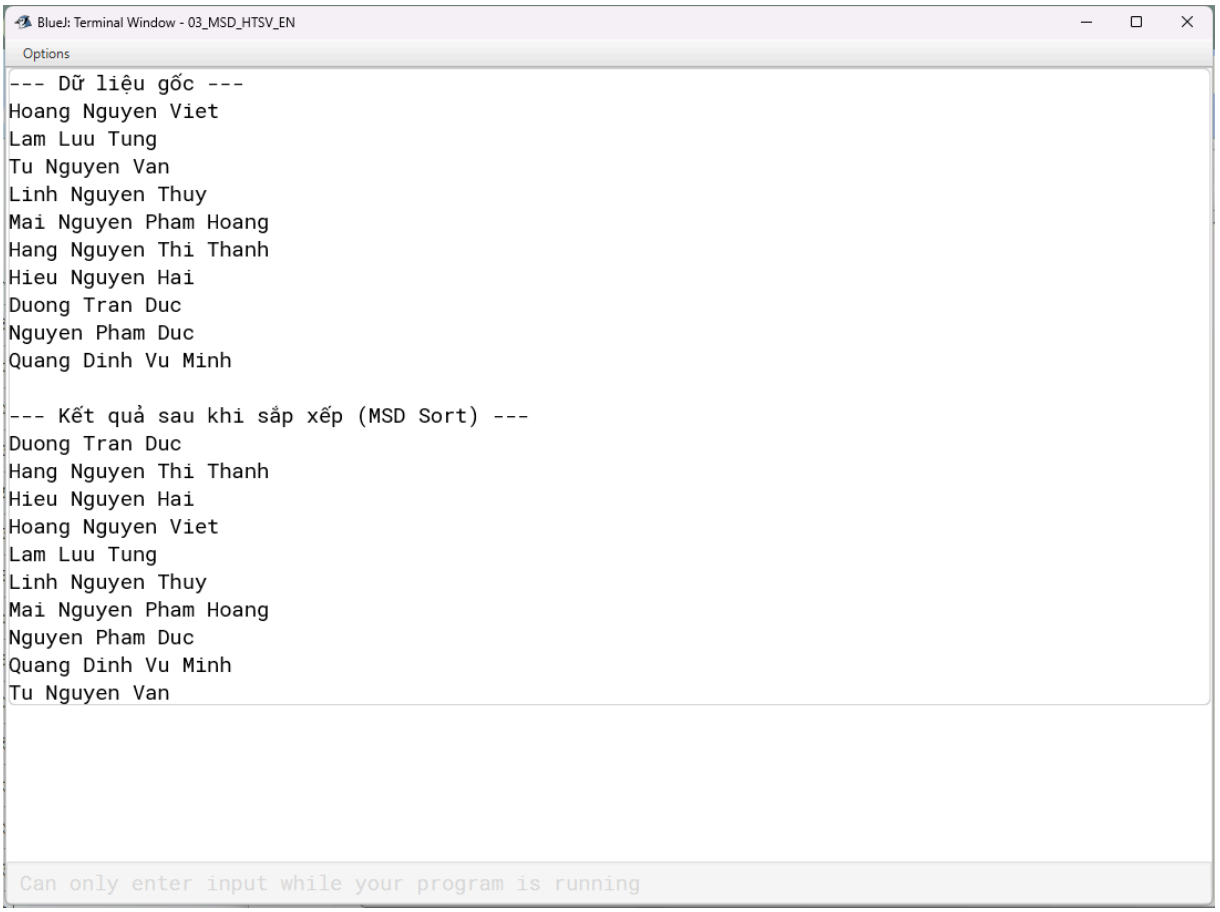
MSD.java

```
public class MSD {
    private static final int BITS_PER_BYTE =   8;
    private static final int BITS_PER_INT  =  32;
    private static final int R             = 256;
    private static final int CUTOFF        =  15;
    private MSD() { }
    public static void sort(String[] a) {
        int n = a.length;
        String[] aux = new String[n];
        sort(a, 0, n-1, 0, aux);
    }
    private static int charAt(String s, int d) {
        if (d == s.length()) return -1;
        return s.charAt(d);
    }
```

```
    private static void sort(String[] a, int lo, int hi, int d, String[] aux) {
        if (hi <= lo + CUTOFF) {
            insertion(a, lo, hi, d);
            return;
        }
        int[] count = new int[R+2];
        for (int i = lo; i <= hi; i++) {
            int c = charAt(a[i], d);
            count[c+2]++;
        }
        for (int r = 0; r < R+1; r++)
            count[r+1] += count[r];
        for (int i = lo; i <= hi; i++) {
            int c = charAt(a[i], d);
            aux[count[c+1]++] = a[i];
        }
        for (int i = lo; i <= hi; i++)
            a[i] = aux[i - lo];
        for (int r = 0; r < R; r++)
            sort(a, lo + count[r], lo + count[r+1] - 1, d+1, aux);
    }
    private static void insertion(String[] a, int lo, int hi, int d) {
        for (int i = lo; i <= hi; i++)
            for (int j = i; j > lo && less(a[j], a[j-1], d); j--)
                exch(a, j, j-1);
    }
    private static void exch(String[] a, int i, int j) {
        String temp = a[i];
        a[i] = a[j];
        a[j] = temp;
    }
    private static boolean less(String v, String w, int d) {
        for (int i = d; i < Math.min(v.length(), w.length()); i++) {
            if (v.charAt(i) < w.charAt(i)) return true;
            if (v.charAt(i) > w.charAt(i)) return false;
        }
        return v.length() < w.length();
    }
    public static void sort(int[] a) {
        int n = a.length;
        int[] aux = new int[n];
        sort(a, 0, n-1, 0, aux);
    }
    private static void sort(int[] a, int lo, int hi, int d, int[] aux) {
        if (hi <= lo + CUTOFF) {
            insertion(a, lo, hi, d);
            return;
        }

        int[] count = new int[R+1];
        int mask = R - 1;
        int shift = BITS_PER_INT - BITS_PER_BYTE*d - BITS_PER_BYTE;
```

```
        for (int i = lo; i <= hi; i++) {
            int c = (a[i] >> shift) & mask;
            count[c + 1]++;
        }

        for (int r = 0; r < R; r++)
            count[r+1] += count[r];

        for (int i = lo; i <= hi; i++) {
            int c = (a[i] >> shift) & mask;
            aux[count[c]++] = a[i];
        }

        for (int i = lo; i <= hi; i++)
            a[i] = aux[i - lo];

        if (d == 4) return;

        if (count[0] > 0)
            sort(a, lo, lo + count[0] - 1, d+1, aux);

        for (int r = 0; r < R; r++)
            if (count[r+1] > count[r])
                sort(a, lo + count[r], lo + count[r+1] - 1, d+1, aux);
    }

    private static void insertion(int[] a, int lo, int hi, int d) {
        for (int i = lo; i <= hi; i++)
            for (int j = i; j > lo && a[j] < a[j-1]; j--)
                exch(a, j, j-1);
    }

    private static void exch(int[] a, int i, int j) {
        int temp = a[i];
        a[i] = a[j];
        a[j] = temp;
    }
}
```

Kết quả:



Bài tập 4: Ứng dụng MSD Radix Sort để sắp xếp danh sách tên Tiếng Việt

Code:

VietnameseAlphabet.java

```
public class VietnameseAlphabet extends Alphabet {
    public static final VietnameseAlphabet VIETNAMESE_ALPHABET = new
VietnameseAlphabet();

    public VietnameseAlphabet() {
        super(
            "\t\n\r \u00A0" +
            "!\"#$%&'()*+,-./:;<=>?@[]^_`{|}~\\" +
            "0123456789" +
            "AÀÁẢÃẠ" + "aàáảãạ" +
            "ĂẰẮẲẴẶ" + "ăằắẳẵặ" +
            "ÂẦẤẨẪẬ" + "âầấẩẫậ" +
            "B" + "b" +
            "C" + "c" +
            "D" + "d" +
            "Đ" + "đ" +
            "EÈÉẺẼẸ" + "eèéẻẽẹ" +
            "ÊỀẾỂỄỆ" + "êềếểễệ" +
            "F" + "f" +
            "G" + "g" +
            "H" + "h" +
            "IÌÍỈĨỊ" + "iìíỉĩị" +
```

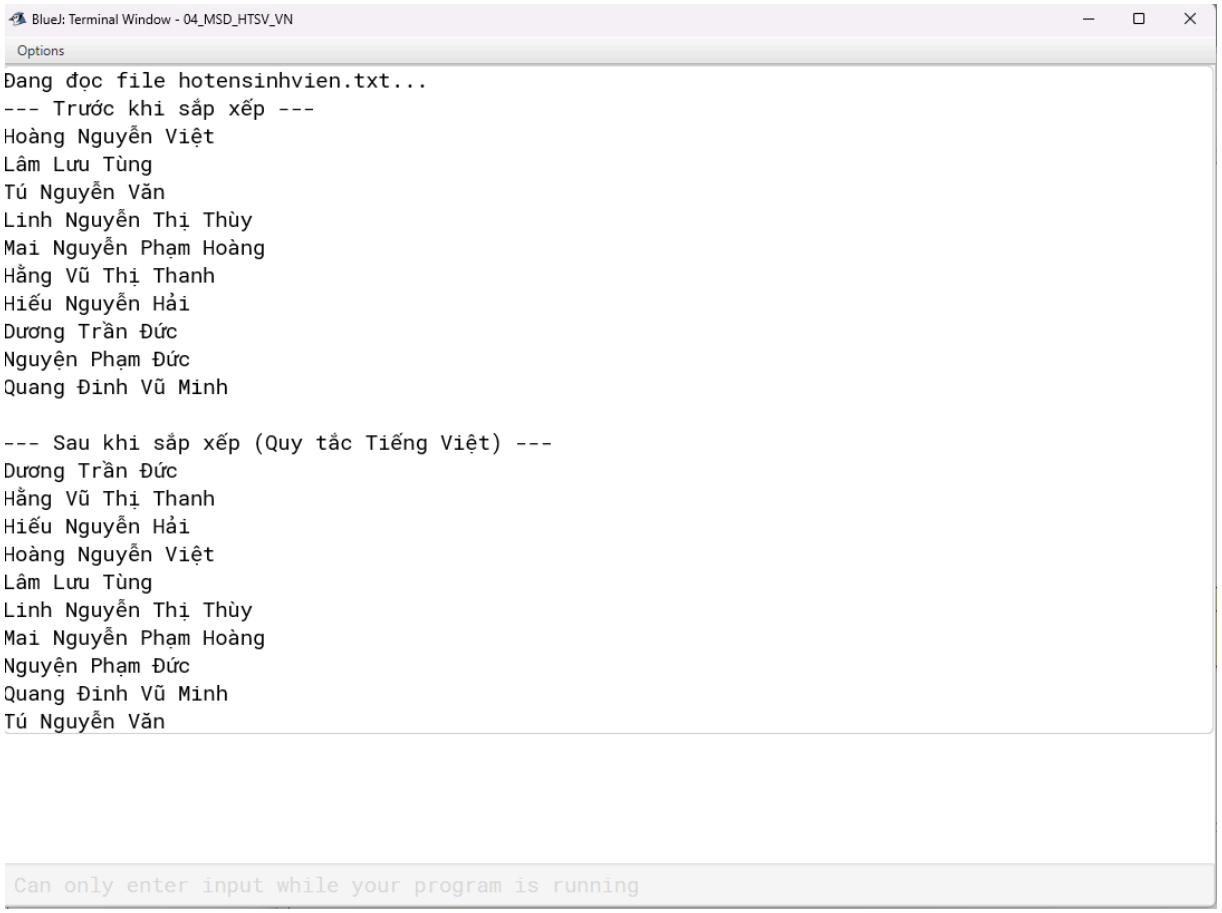
```
            "J" + "j" +
            "K" + "k" +
            "L" + "l" +
            "M" + "m" +
            "N" + "n" +
            "OÒÓỎÕỌ" + "oòóỏõọ" +
            "ÔỒỐỔỖỘ" + "ôồốổỗộ" +
            "ƠỜỚỞỠỢ" + "ơờớởỡợ" +
            "P" + "p" +
            "Q" + "q" +
            "R" + "r" +
            "S" + "s" +
            "T" + "t" +
            "UÙÚỦŨỤ" + "uùúủũụ" +
            "ƯỪỨỬỮỰ" + "ưừứửữự" +
            "V" + "v" +
            "W" + "w" +
            "X" + "x" +
            "YỲÝỶỸỴ" + "yỳýỷỹỵ" +
            "Z" + "z"
        );
    }
}
```

MSD.java

```
public class MSD {
    private static final int R = VietnameseAlphabet.VIETNAMESE_ALPHABET.radix();
    private static final int CUTOFF = 15;

    private MSD() {}
    public static void sort(String[] a) {
        int n = a.length;
        String[] aux = new String[n];
        sort(a, 0, n - 1, 0, aux);
    }
    private static int charAt(String s, int d) {
        if (d >= s.length()) return -1;
        char c = s.charAt(d);
        if (VietnameseAlphabet.VIETNAMESE_ALPHABET.contains(c)) {
            return VietnameseAlphabet.VIETNAMESE_ALPHABET.toIndex(c);
        } else {

            return R;
        }
    }

    private static void sort(String[] a, int lo, int hi, int d, String[] aux) {
        if (hi <= lo + CUTOFF) {
            insertion(a, lo, hi, d);
            return;
        }

        int[] count = new int[R + 2];
```

```
        for (int i = lo; i <= hi; i++) {
            int c = charAt(a[i], d);
            count[c + 2]++;
        }

        for (int r = 0; r < R + 1; r++) {
            count[r + 1] += count[r];
        }

        for (int i = lo; i <= hi; i++) {
            int c = charAt(a[i], d);
            aux[count[c + 1]++] = a[i];
        }

        for (int i = lo; i <= hi; i++) {
            a[i] = aux[i - lo];
        }

        for (int r = 0; r < R; r++) {
            sort(a, lo + count[r], lo + count[r + 1] - 1, d + 1, aux);
        }
    }
    private static void insertion(String[] a, int lo, int hi, int d) {
        for (int i = lo; i <= hi; i++) {
            for (int j = i; j > lo && less(a[j], a[j - 1], d); j--) {
                exch(a, j, j - 1);
            }
        }
    }

    private static boolean less(String v, String w, int d) {
        for (int i = d; i < Math.min(v.length(), w.length()); i++) {
            char vc = v.charAt(i);
            char wc = w.charAt(i);

            int orderV = VietnameseAlphabet.VIETNAMESE_ALPHABET.contains(vc) ?
                         VietnameseAlphabet.VIETNAMESE_ALPHABET.toIndex(vc) : R;
            int orderW = VietnameseAlphabet.VIETNAMESE_ALPHABET.contains(wc) ?
                         VietnameseAlphabet.VIETNAMESE_ALPHABET.toIndex(wc) : R;

            if (orderV < orderW) return true;
            if (orderV > orderW) return false;
        }
        return v.length() < w.length();
    }

    private static void exch(String[] a, int i, int j) {
        String temp = a[i];
        a[i] = a[j];
        a[j] = temp;
    }
}
```

Kết quả:



Bài tập 5: Ứng dụng Quick3Way String Sort và Suffix Array cho Tiếng Việt

Phần A: Sắp xếp danh sách sinh viên bằng Quick3Way String Sort
Phần B: Xây dựng Suffix Array cho văn bản Tiếng Việt
Code:

Quick3string.java

```java
import java.util.Random;

public class Quick3string {
    protected static final int CUTOFF = 15;
    protected final Alphabet alphabet;

    public Quick3string(Alphabet alphabet) {
        this.alphabet = alphabet;
    }

    public void sort(String[] a) {
        Random rand = new Random();
        for (int i = 0; i < a.length; i++) {
            int r = i + rand.nextInt(a.length - i);
            String temp = a[i]; a[i] = a[r]; a[r] = temp;
        }
        sort(a, 0, a.length - 1, 0);
    }
    protected int charAt(String s, int d) {
        if (d >= s.length()) return -1;
```

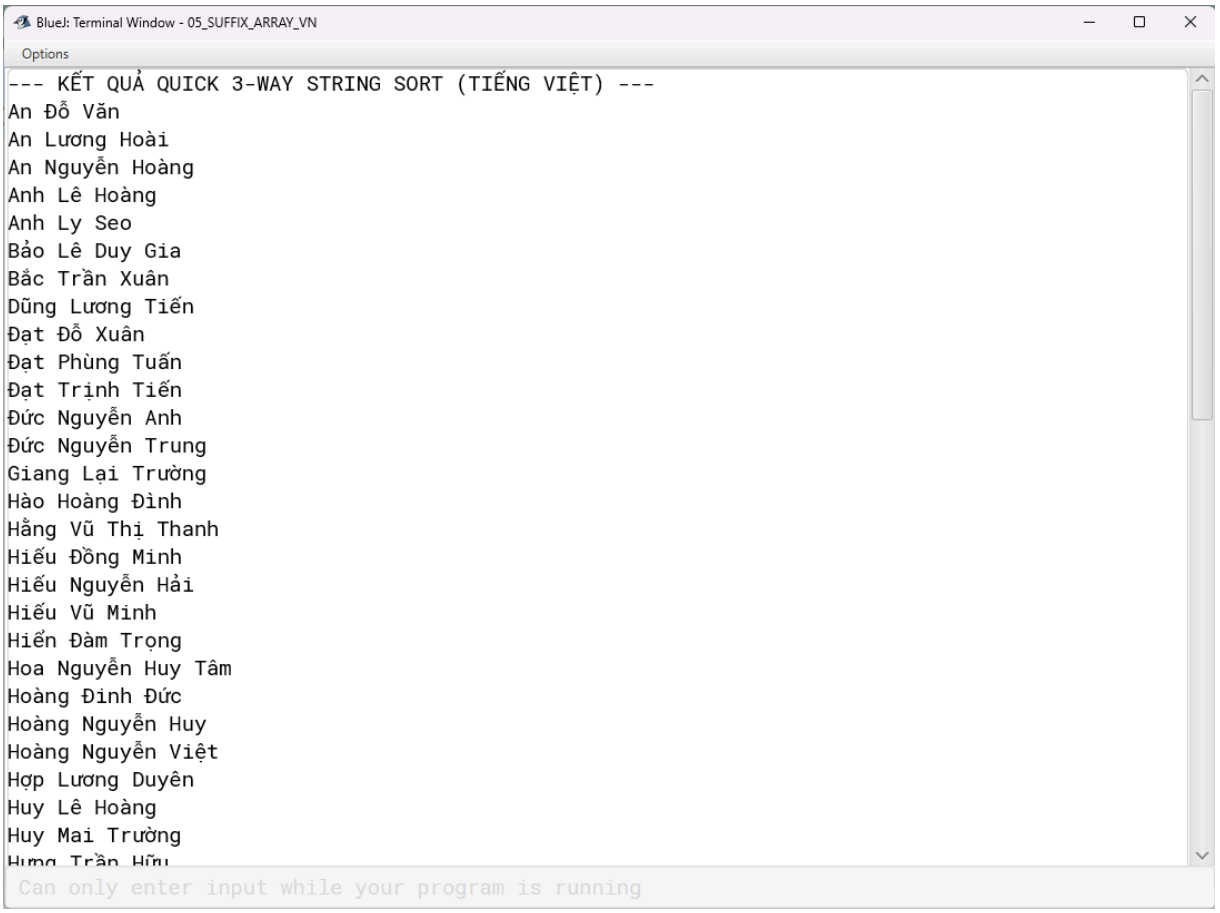
```
        return s.charAt(d);
    }
    protected void sort(String[] a, int lo, int hi, int d) {
        if (hi <= lo + CUTOFF) {
            insertion(a, lo, hi, d);
            return;
        }

        int lt = lo, gt = hi;
        int v = charAt(a[lo], d);
        int i = lo + 1;
        while (i <= gt) {
            int t = charAt(a[i], d);
            if (t < v) exch(a, lt++, i++);
            else if (t > v) exch(a, i, gt--);
            else i++;
        }

        sort(a, lo, lt - 1, d);
        if (v >= 0) sort(a, lt, gt, d + 1);
        sort(a, gt + 1, hi, d);
    }
    protected void insertion(String[] a, int lo, int hi, int d) {
        for (int i = lo; i <= hi; i++)
            for (int j = i; j > lo && less(a[j], a[j - 1], d); j--)
                exch(a, j, j - 1);
    }
    protected void exch(String[] a, int i, int j) {
        String temp = a[i]; a[i] = a[j]; a[j] = temp;
    }
    protected boolean less(String v, String w, int d) {
        for (int i = d; i < Math.min(v.length(), w.length()); i++) {
            if (charAt(v, i) < charAt(w, i)) return true;
            if (charAt(v, i) > charAt(w, i)) return false;
        }
        return v.length() < w.length();
    }
}
```

Quick3stringVietnamese.java

```
public class Quick3stringVietnamese extends Quick3string {
    public Quick3stringVietnamese() {
        super(VietnameseAlphabet.VIETNAMESE_ALPHABET);
    }
    @Override
    protected int charAt(String s, int d) {
        if (d >= s.length()) return -1;
        char c = s.charAt(d);
        return alphabet.contains(c) ? alphabet.toIndex(c) : -1;
    }
    @Override
    protected boolean less(String v, String w, int d) {
        for (int i = d; i < Math.min(v.length(), w.length()); i++) {
```

```
            int vIndex = charAt(v, i);
            int wIndex = charAt(w, i);
            if (vIndex < wIndex) return true;
            if (vIndex > wIndex) return false;
        }
        return v.length() < w.length();
    }
}
```

Kết quả:



SuffixArrayX.java

```
public class SuffixArrayX {
    private static final int CUTOFF = 5;

    protected final char[] text;
    protected final int[] index;
    protected final int n;
    protected final Alphabet alphabet;

    public SuffixArrayX(String text, Alphabet alphabet) {
        this.alphabet = alphabet;
        n = text.length();
        String s = text + '\0';
        this.text = s.toCharArray();
        this.index = new int[n];
        for (int i = 0; i < n; i++)
            index[i] = i;
```

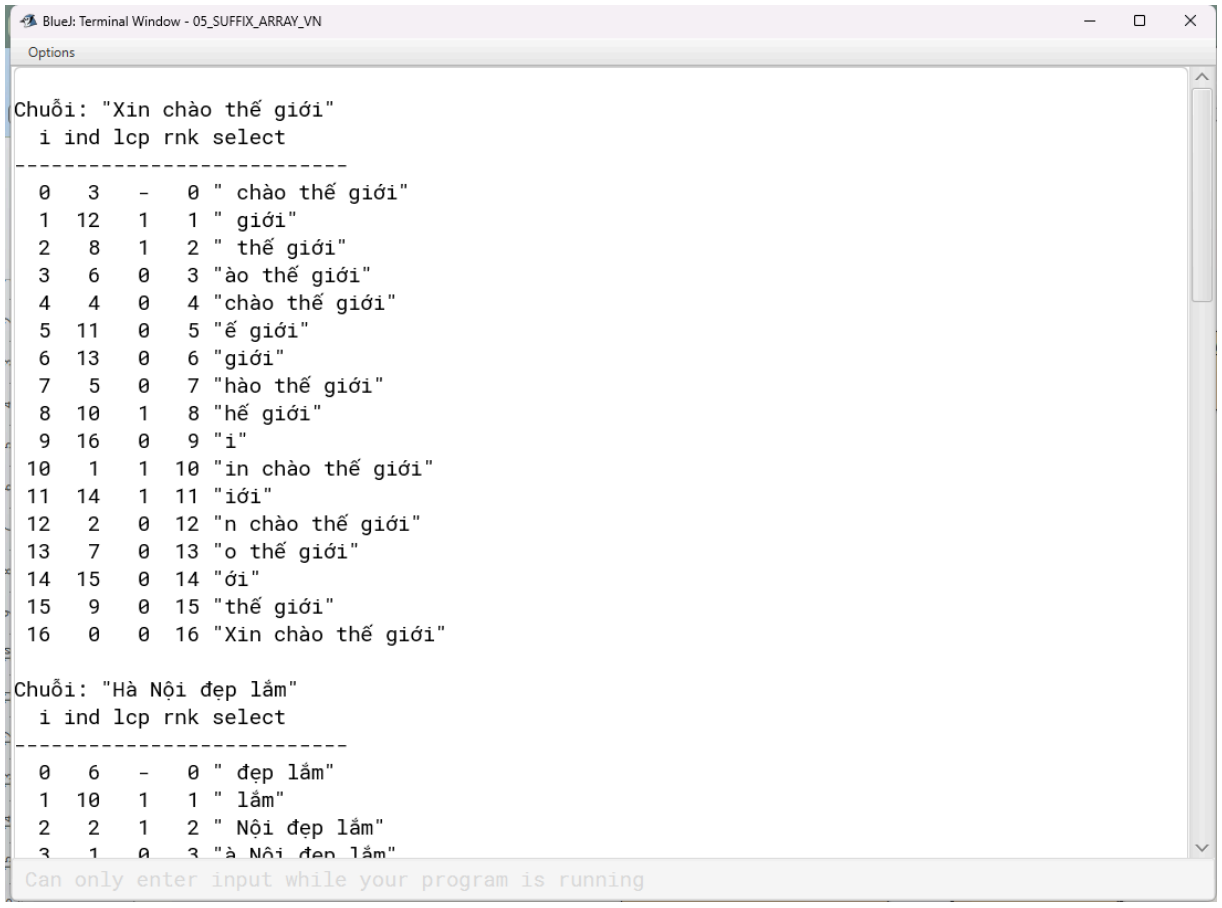
```
        sort(0, n - 1, 0);
    }
    protected int charAt(int pos, int offset) {
        char c = text[pos + offset];
        if (c == '\0') return -1;
        if (alphabet == null) return c;
        return alphabet.toIndex(c);
    }
    protected void sort(int lo, int hi, int d) {
        if (hi <= lo + CUTOFF) {
            insertion(lo, hi, d);
            return;
        }

        int lt = lo, gt = hi;
        int v = charAt(index[lo], d);
        int i = lo + 1;
        while (i <= gt) {
            int t = charAt(index[i], d);
            if (t < v) exch(lt++, i++);
            else if (t > v) exch(i, gt--);
            else i++;
        }

        sort(lo, lt - 1, d);
        if (v > 0) sort(lt, gt, d + 1);
        sort(gt + 1, hi, d);
    }

    protected void insertion(int lo, int hi, int d) {
        for (int i = lo; i <= hi; i++)
            for (int j = i; j > lo && less(index[j], index[j - 1], d); j--)
                exch(j, j - 1);
    }

    protected boolean less(int i, int j, int d) {
        if (i == j) return false;
        i += d; j += d;
        while (i < n && j < n) {
            int ci = charAt(i, 0);
            int cj = charAt(j, 0);
            if (ci < cj) return true;
            if (ci > cj) return false;
            i++; j++;
        }
        return i > j;
    }

    protected void exch(int i, int j) {
        int swap = index[i]; index[i] = index[j]; index[j] = swap;
    }

    public int length() { return n; }
```

```
    public int index(int i) { return index[i]; }
    public int lcp(int i) {
        if (i < 1 || i >= n) throw new IllegalArgumentException();
        return lcp(index[i], index[i - 1]);
    }

    protected int lcp(int i, int j) {
        int length = 0;
        while (i < n && j < n) {
            if (text[i] != text[j]) return length;
            i++; j++; length++;
        }
        return length;
    }
    public int rank(String query) {
        int lo = 0, hi = n - 1;
        while (lo <= hi) {
            int mid = lo + (hi - lo) / 2;
            int cmp = compare(query, index[mid]);
            if (cmp < 0) hi = mid - 1;
            else if (cmp > 0) lo = mid + 1;
            else return mid;
        }
        return lo;
    }

    protected int compare(String query, int i) {
        int m = query.length();
        int j = 0;
        while (i < n && j < m) {
            int qc = (alphabet == null) ? query.charAt(j) :
alphabet.toIndex(query.charAt(j));
            int tc = charAt(i, 0);
            if (qc != tc) return qc - tc;
            i++; j++;
        }
        if (i < n) return -1;
        if (j < m) return +1;
        return 0;
    }
}
```

SuffixArrayXVN.java

```
public class SuffixArrayXVN extends SuffixArrayX {
    public SuffixArrayXVN(String text) {
        super(text, VietnameseAlphabet.VIETNAMESE_ALPHABET);
    }
}
```

Kết quả:



Bài tập 6: Tìm kiếm từ khóa trong ngữ cảnh (Keyword-in-Context - KWIC) cho văn bản Tiếng Việt

Code:

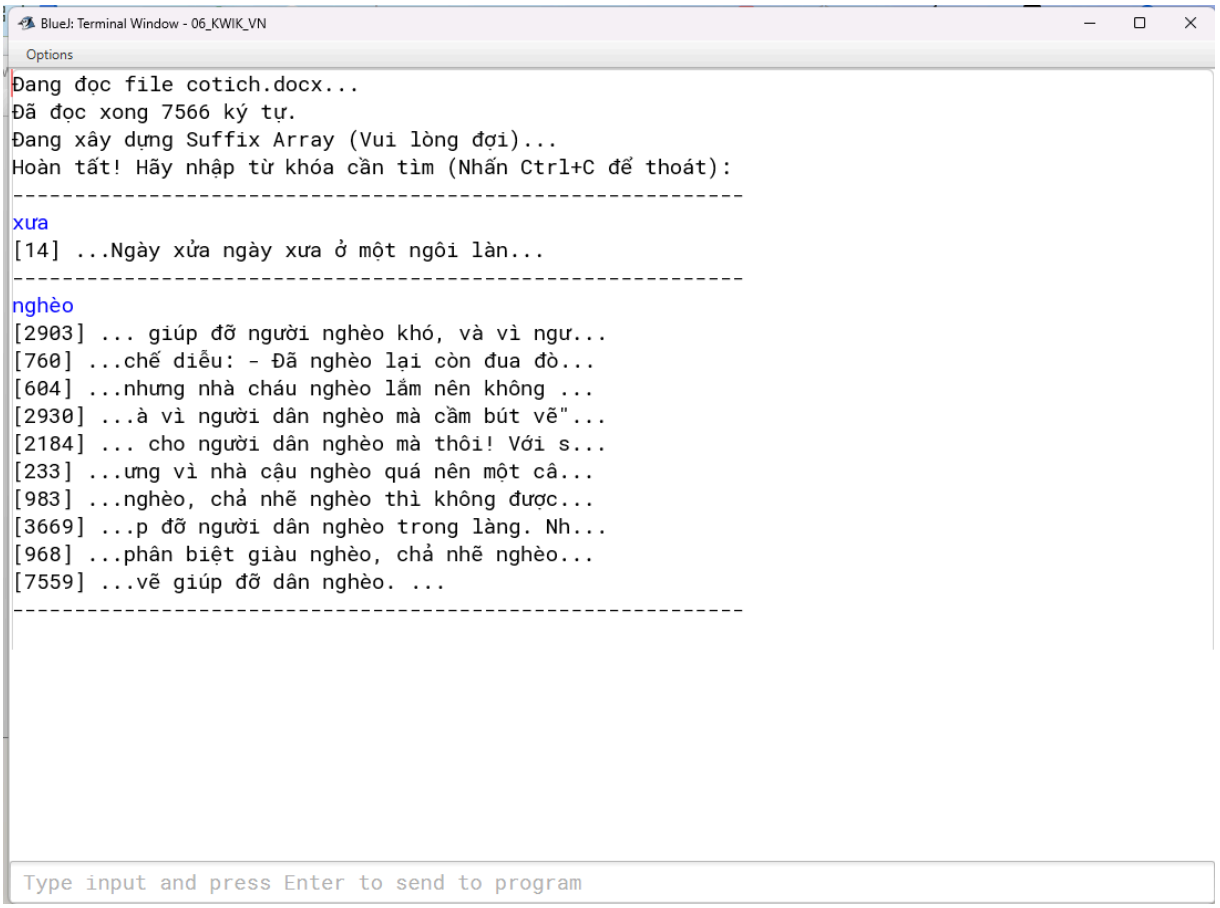
KWIK.java

```
import java.io.*;
import java.util.*;
import org.apache.poi.xwpf.usermodel.XWPFDocument;
import org.apache.poi.xwpf.usermodel.XWPFParagraph;

public class KWIK {
    private KWIK() { }

    public static void main(String[] args) {
        if (args.length < 2) {
            System.err.println("Cách dùng: java KWIK <file.docx>
<độ_dài_ngữ_cảnh>");
            return;
        }

        String filePath = args[0];
        int context;
        try {
            context = Integer.parseInt(args[1]);
        } catch (NumberFormatException e) {
            System.err.println("Lỗi: Tham số thứ hai phải là một số nguyên (ví
dụ: 20).");
            return;
```

```
        }
        System.out.println("Đang đọc file " + filePath + "...");
        StringBuilder textBuilder = new StringBuilder();

        try (FileInputStream fis = new FileInputStream(new File(filePath));
             XWPFDocument document = new XWPFDocument(fis)) {
            List<XWPFParagraph> paragraphs = document.getParagraphs();
            for (XWPFParagraph para : paragraphs) {
                String paraText = para.getText().trim();
                if (!paraText.isEmpty()) {
                    textBuilder.append(paraText).append(" ");
                }
            }
        } catch (IOException e) {
            System.err.println("Lỗi khi đọc file .docx: " + e.getMessage());
            return;
        }
        String text = textBuilder.toString().replaceAll("\\s+", " ");
        int n = text.length();

        System.out.println("Đã đọc xong " + n + " ký tự.");
        System.out.println("Đang xây dựng Suffix Array (Vui lòng đợi)...");
        SuffixArrayXVN sa = new SuffixArrayXVN(text);

        System.out.println("Hoàn tất! Hãy nhập từ khóa cần tìm (Nhấn Ctrl+C để
thoát):");

System.out.println("------------------------------------------------------------"
);
        while (StdIn.hasNextLine()) {
            String query = StdIn.readLine().trim();
            if (query.isEmpty()) continue;

            boolean found = false;
            for (int i = sa.rank(query); i < n; i++) {
                int from1 = sa.index(i);
                int to1 = Math.min(n, from1 + query.length());
                if (!query.equals(text.substring(from1, to1))) break;

                found = true;

                int from2 = Math.max(0, sa.index(i) - context);
                int to2 = Math.min(n, sa.index(i) + context + query.length());

                StdOut.println("[" + from1 + "] ..." + text.substring(from2,
to2) + "...");
            }

            if (!found) {
                StdOut.println("Không tìm thấy từ khóa: \"" + query + "\"");
            }

StdOut.println("------------------------------------------------------------");
```

```
        }
    }
}
```

Kết quả:



Bài tập 7: Ứng dụng Suffix Array tìm đoạn mã lặp (Code Duplication Detection)

Code:

program.c

```c
#include <stdio.h>

void multiplyMatrix(int A[2][2], int B[2][2], int C[2][2]) {
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            C[i][j] = 0;
            for (int k = 0; k < 2; k++) {
                C[i][j] += A[i][k] * B[k][j];
            }
        }
    }
}

int main() {
    int A[2][2] = {{1, 2}, {3, 4}};
    int B[2][2] = {{5, 6}, {7, 8}};
    int C[2][2];
```

```
    multiplyMatrix(A, B, C);

    int D[2][2];
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            D[i][j] = 0;
            for (int k = 0; k < 2; k++) {
                D[i][j] += A[i][k] * B[k][j];
            }
        }
    }

    int E[2][2];
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            E[i][j] = 0;
            for (int k = 0; k < 2; k++) {
                E[i][j] += A[i][k] * B[k][j];
            }
        }
    }

    int F[2][2];
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            F[i][j] = 0;
            for (int k = 0; k < 2; k++) {
                F[i][j] += A[i][k] * B[k][j];
            }
        }
    }

    return 0;
}
```

LongestRepeatedSubstring.java

```
public class LongestRepeatedSubstring {
    private LongestRepeatedSubstring() { }
    public static String lrs(String text) {
        int n = text.length();
        SuffixArray sa = new SuffixArray(text);

        String lrs = "";
        for (int i = 1; i < n; i++) {
            int length = sa.lcp(i);

            if (length > lrs.length()) {
                lrs = text.substring(sa.index(i), sa.index(i) + length);
            }
        }
        return lrs;
    }
```

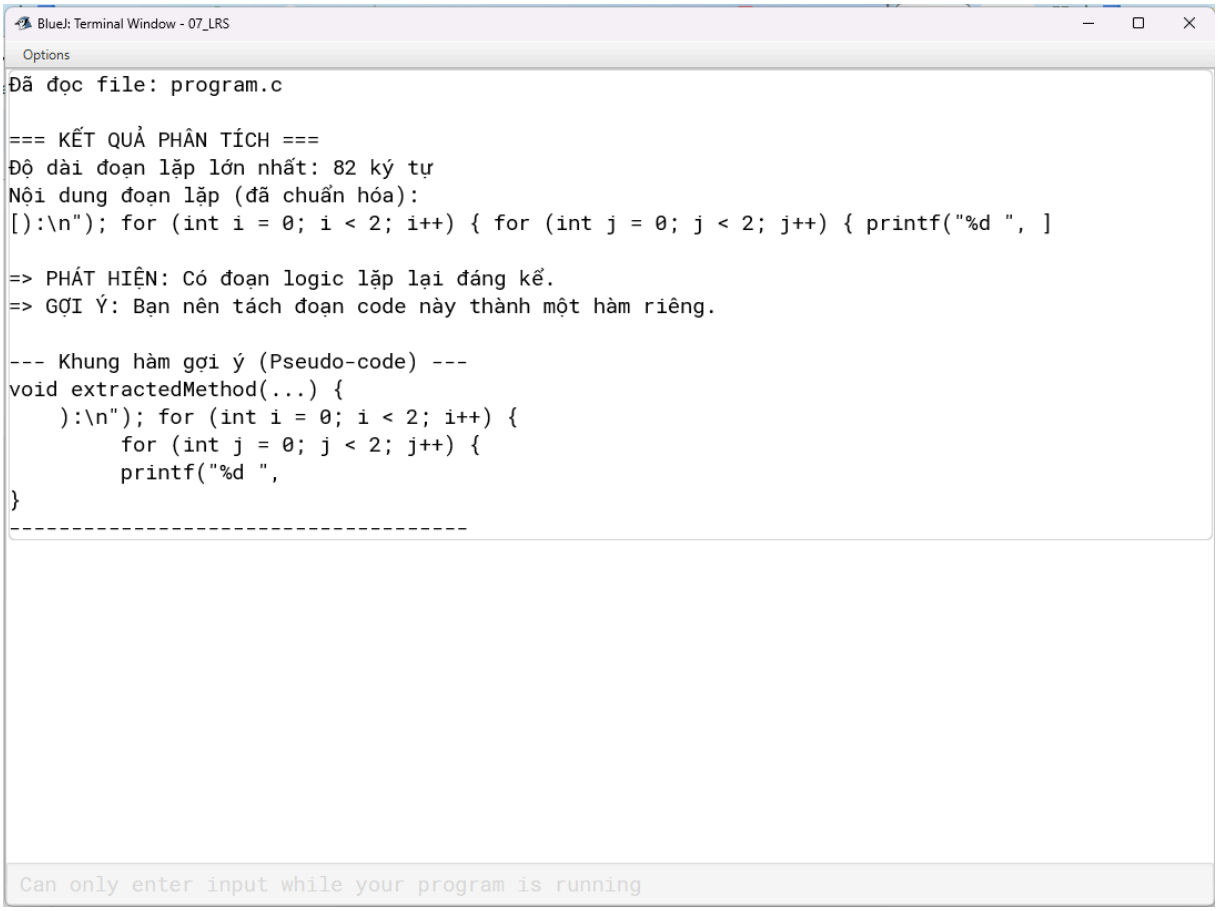
```
}
```

TestCodeRefactoring.java

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
public class TestCodeRefactoring {
    public static void main(String[] args) {
        String fileName = "program.c";
        String text;
        try {
            text = Files.readString(Path.of(fileName));
            System.out.println("Đã đọc file: " + fileName);
        } catch (IOException e) {
            System.out.println("Lỗi: Không thể đọc file " + fileName);
            return;
        }
        String cleanedText = text.replaceAll("\\s+", " ");
        String result = LongestRepeatedSubstring.lrs(cleanedText);

        System.out.println("\n=== KẾT QUẢ PHÂN TÍCH ===");
        System.out.println("Độ dài đoạn lặp lớn nhất: " + result.length() + "
ký tự");

        String displayStr = result.length() > 100 ? result.substring(0, 100) +
"..." : result;
        System.out.println("Nội dung đoạn lặp (đã chuẩn hóa): \n[" + displayStr
+ "]");

        if (isLikelyCodeBlock(result)) {
            System.out.println("\n=> PHÁT HIỆN: Có đoạn logic lặp lại đáng
kể.");
            System.out.println("=> GỢI Ý: Bạn nên tách đoạn code này thành một
hàm riêng.");

            suggestFunctionSkeleton(result);
        } else {
            System.out.println("\n=> Không phát hiện đoạn code lặp nào đủ lớn
hoặc có cấu trúc rõ ràng.");
        }
    }
    private static boolean isLikelyCodeBlock(String substring) {
        return (substring.contains("{") && substring.contains("}"))
            || substring.contains("for")
            || substring.contains("while")
            || substring.contains("if");
    }

    private static void suggestFunctionSkeleton(String substring) {
        System.out.println("\n--- Khung hàm gợi ý (Pseudo-code) ---");
        System.out.println("void extractedMethod(...) {");
        System.out.println("    " + substring.replace("}", "}\n
").replace("{", "{\n        "));
```

```
        System.out.println("}");
        System.out.println("-------------------------------------");
    }
}
```

Kết quả:



Bài tập 8: Tìm xâu con chung dài nhất (LCS) của hai báo cáo Tiếng Việt

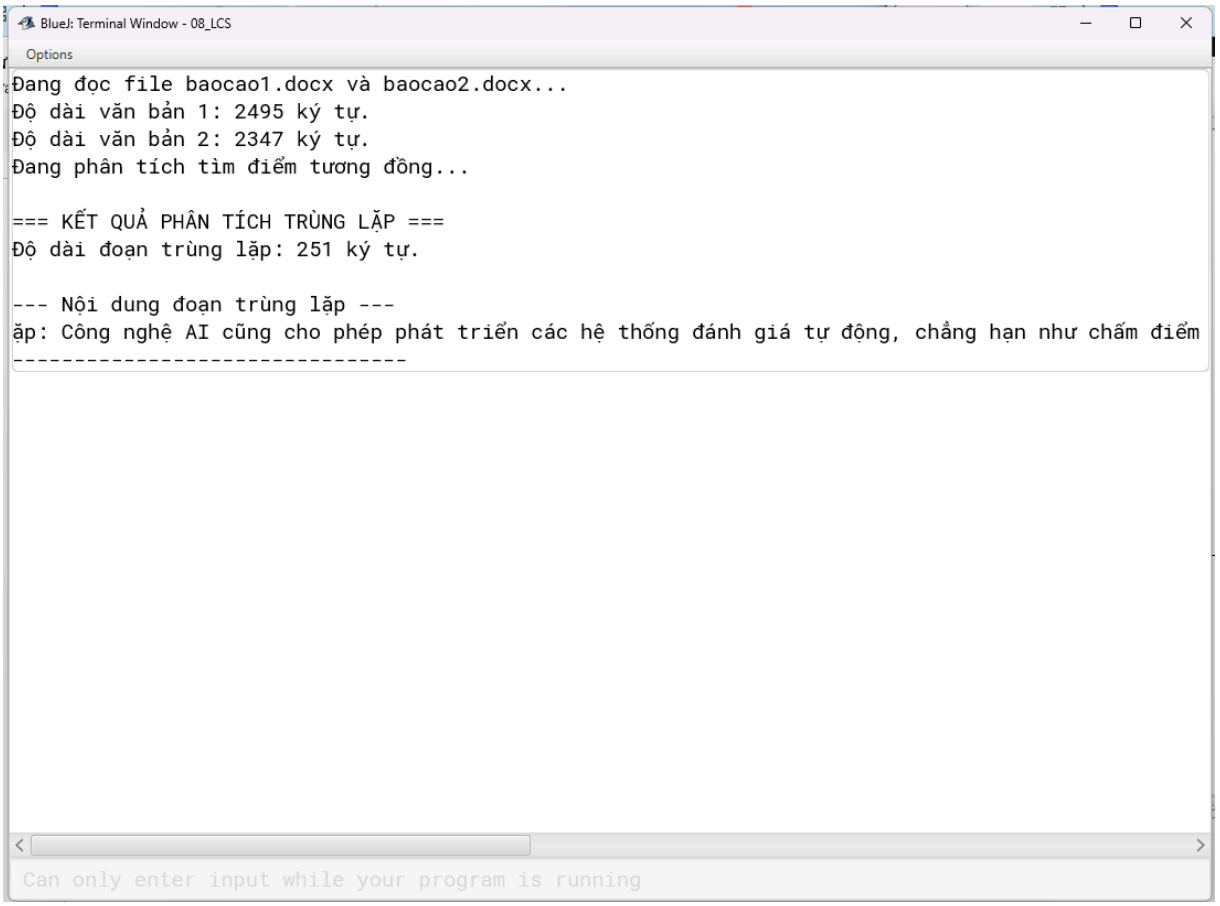
Code:

LongestCommonSubstring.java

```
public class LongestCommonSubstring {
    private LongestCommonSubstring() { }
    private static String lcp(String s, int p, String t, int q) {
        int n = Math.min(s.length() - p, t.length() - q);
        for (int i = 0; i < n; i++) {
            if (s.charAt(p + i) != t.charAt(q + i))
                return s.substring(p, p + i);
        }
        return s.substring(p, p + n);
    }
    private static int compare(String s, int p, String t, int q) {
        int n = Math.min(s.length() - p, t.length() - q);
        for (int i = 0; i < n; i++) {
            if (s.charAt(p + i) != t.charAt(q + i))
                return s.charAt(p + i) - t.charAt(q + i);
        }
        if      (s.length() - p < t.length() - q) return -1;
```

```
        else if (s.length() - p > t.length() - q) return +1;
        else                                      return  0;
    }
    public static String lcs(String s, String t) {
        SuffixArrayXVN suffix1 = new SuffixArrayXVN(s);
        SuffixArrayXVN suffix2 = new SuffixArrayXVN(t);

        String lcs = "";
        int i = 0, j = 0;

        while (i < s.length() && j < t.length()) {
            int p = suffix1.index(i);
            int q = suffix2.index(j);

            String x = lcp(s, p, t, q);

            if (x.length() > lcs.length()) {
                lcs = x;
            }

            if (compare(s, p, t, q) < 0) i++;
            else                         j++;
        }
        return lcs;
    }
}
```

Kết quả:



Bài tập 9: Ứng dụng TrieST (R-way Trie) xây dựng hệ thống gợi ý từ Tiếng Việt

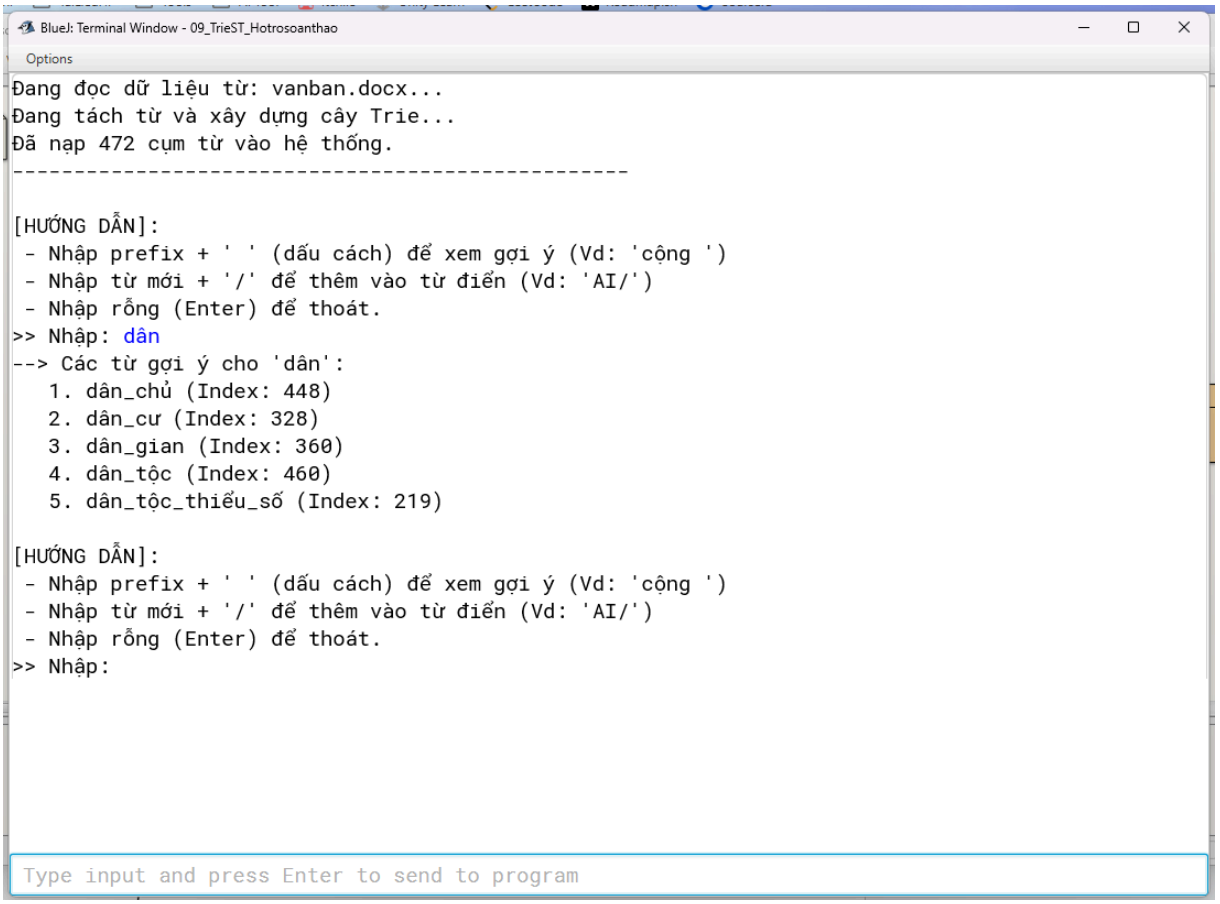
Code:

AutoCompleteSystem.java

```
import vn.pipeline.VnCoreNLP;
import java.util.List;
import java.util.Scanner;
import java.io.File;
import java.io.FileInputStream;
import org.apache.poi.xwpf.usermodel.XWPFDocument;
import org.apache.poi.xwpf.usermodel.XWPFParagraph;

public class AutocompleteSystem {

    public static void main(String[] args) throws Exception {
        String[] annotators = {"wseg"};
        VnCoreNLP pipeline = new VnCoreNLP(annotators);
        String filePath = "vanban.docx";
        System.out.println("Đang đọc dữ liệu từ: " + filePath + "...");

        StringBuilder textBuilder = new StringBuilder();
        try (FileInputStream fis = new FileInputStream(new File(filePath));
             XWPFDocument document = new XWPFDocument(fis)) {
            for (XWPFParagraph para : document.getParagraphs()) {
                String text = para.getText().trim();
                if (!text.isEmpty()) {
                    textBuilder.append(text).append(" ");
                }
            }
        } catch (Exception e) {
            System.err.println("Lỗi đọc file: " + e.getMessage());
            return;
        }

        String inputText = textBuilder.toString().trim().replaceAll("\\s+", "
");

        System.out.println("Đang tách từ và xây dựng cây Trie...");

        List<String> words = TextProcessor.segmentText(inputText);

        TrieST<Integer> trie = new
TrieST<>(VietnameseAlphabet.VIETNAMESE_ALPHABET);

        for (int i = 0; i < words.size(); i++) {
            String word = words.get(i);
            trie.put(word, i);
        }

        System.out.println("Đã nạp " + words.size() + " cụm từ vào hệ thống.");

System.out.println("-------------------------------------------------");

        Scanner scanner = new Scanner(System.in, "UTF-8");
```

```
        int nextPosition = words.size(); // Dùng để đánh số cho từ mới thêm vào

        while (true) {
            System.out.println("\n[HƯỚNG DẪN]:");
            System.out.println(" - Nhập prefix + ' ' (dấu cách) để xem gợi ý
(Vd: 'cộng ')");
            System.out.println(" - Nhập từ mới + '/' để thêm vào từ điển (Vd:
'AI/')");
            System.out.println(" - Nhập rỗng (Enter) để thoát.");
            System.out.print(">> Nhập: ");

            String input = scanner.nextLine();
            if (input.isEmpty()) break;

            if (input.endsWith(" ")) {
                String prefix = input.trim();
                System.out.println("--> Các từ gợi ý cho '" + prefix + "':");
                Iterable<String> suggestions = trie.keysWithPrefix(prefix);

                int count = 0;
                for (String s : suggestions) {
                    Integer pos = trie.get(s);
                    System.out.println("   " + (++count) + ". " + s + " (Index:
" + pos + ")");
                }

                if (count == 0) {
                    System.out.println("   (Không tìm thấy từ nào)");
                }

            } else if (input.endsWith("/")) {
                String newWord = input.substring(0, input.length() - 1).trim();

                if (!trie.contains(newWord)) {
                    trie.put(newWord, nextPosition++);
                    System.out.println("--> Đã thêm từ mới: [" + newWord +
"]");
                } else {
                    System.out.println("--> Từ [" + newWord + "] đã tồn tại
trong hệ thống.");
                }
            } else {
                System.out.println("(!) Vui lòng kết thúc bằng dấu cách (để tìm)
hoặc dấu / (để thêm).");
            }
        }
        scanner.close();
    }
}
```

Kết quả:



Bài tập 10: Quản lý điểm sinh viên bằng cấu trúc cây TST hỗ trợ Tiếng Việt

Code:

```
import java.io.*;
import org.apache.poi.xwpf.usermodel.XWPFDocument;
import org.apache.poi.xwpf.usermodel.XWPFParagraph;
import java.util.ArrayList;
import java.util.List;

public class TestVNTST {

    private TestVNTST() { }

    public static void main(String[] args) {
        String filePath = args.length > 0 ? args[0] : "dssv.docx";

        VietnameseTST<Double> studentTST = new VietnameseTST<Double>();

        List<String> names = new ArrayList<>();

        System.out.println("Đang đọc dữ liệu từ file: " + filePath + "...");

        try (FileInputStream fis = new FileInputStream(new File(filePath));
             XWPFDocument document = new XWPFDocument(fis)) {

            for (XWPFParagraph para : document.getParagraphs()) {
```

```
                String line = para.getText().trim();

                if (!line.isEmpty() && !line.startsWith("[")) {
                    String[] parts = line.split("\\s+(?=[0-9])");

                    if (parts.length >= 2) {
                        String name = parts[0].trim();
                        try {
                            Double grade = Double.parseDouble(parts[1].trim());

                            names.add(name);

                            studentTST.put(name, grade);

                        } catch (NumberFormatException e) {
                            System.err.println("Lỗi định dạng điểm số tại dòng: "
+ line);
                        } catch (IllegalArgumentException e) {
                            System.err.println("Tên chứa ký tự không hợp lệ: "
+ name);
                        }
                    }
                }
            }
        } catch (IOException e) {
            System.err.println("Không thể đọc file " + filePath + ": " +
e.getMessage());
            return;
        }

        System.out.println("Đã nạp thành công " + studentTST.size() + " sinh
viên vào hệ thống.\n");
        System.out.println("=== DANH SÁCH SINH VIÊN (Sắp xếp A-Z Tiếng Việt)
===");
        System.out.printf("%-30s | %s\n", "Họ và Tên", "Điểm TB");
        System.out.println("-----------------------------------------");

        for (String name : studentTST.keys()) {
            System.out.printf("%-30s | %.2f\n", name, studentTST.get(name));
        }
        System.out.println("-----------------------------------------\n");

        if (!names.isEmpty()) {
            String target = names.get(0);
            System.out.println("[Tra cứu] Điểm của '" + target + "': " +
studentTST.get(target));
        }

        String prefix = "Nguyễn";
        System.out.println("\n[Tìm kiếm] Các sinh viên họ '" + prefix + "':");
        for (String name : studentTST.keysWithPrefix(prefix)) {
            System.out.println(" - " + name + ": " + studentTST.get(name));
        }
```

```
        if (!names.isEmpty()) {
            String query = names.get(0) + " Lớp K65";
            String match = studentTST.longestPrefixOf(query);
            System.out.println("\n[Longest Prefix] Tìm thấy tên trong chuỗi '" +
query + "': " + match);
        }
    }
}
```

Kết quả:

```
-------------------------------------------

[Tra cứu] Điểm của 'Đỗ Văn An': 7.92

[Tìm kiếm] Các sinh viên họ 'Nguyễn':
 - Nguyễn Anh Đức: 9.42
 - Nguyễn Bá Nam: 8.03
 - Nguyễn Duy Anh: 8.39
 - Nguyễn Duy Thái: 7.68
 - Nguyễn Hải Hiếu: 8.54
 - Nguyễn Hoàng An: 8.41
 - Nguyễn Huy Hoàng: 8.11
 - Nguyễn Huy Tâm Hoa: 9.87
 - Nguyễn Hữu Tuân: 9.37
 - Nguyễn Phạm Hoàng Mai: 9.91
 - Nguyễn Quốc Thịnh: 9.48
 - Nguyễn Sỹ Quyền: 8.91
 - Nguyễn Thị Thùy Linh: 9.33
 - Nguyễn Tiến Mạnh: 8.42
 - Nguyễn Trung Đức: 8.21
 - Nguyễn Trung Thành: 8.12
 - Nguyễn Văn Tâm: 8.3
 - Nguyễn Văn Thái: 9.23
 - Nguyễn Văn Tú: 8.74
 - Nguyễn Việt Hoàng: 9.79

[Longest Prefix] Tìm thấy tên trong chuỗi 'Đỗ Văn An Lớp K65': Đỗ Văn An
```

Bài tập 11: Quản lý Bảng điểm Sinh viên bằng cấu trúc TST lồng nhau

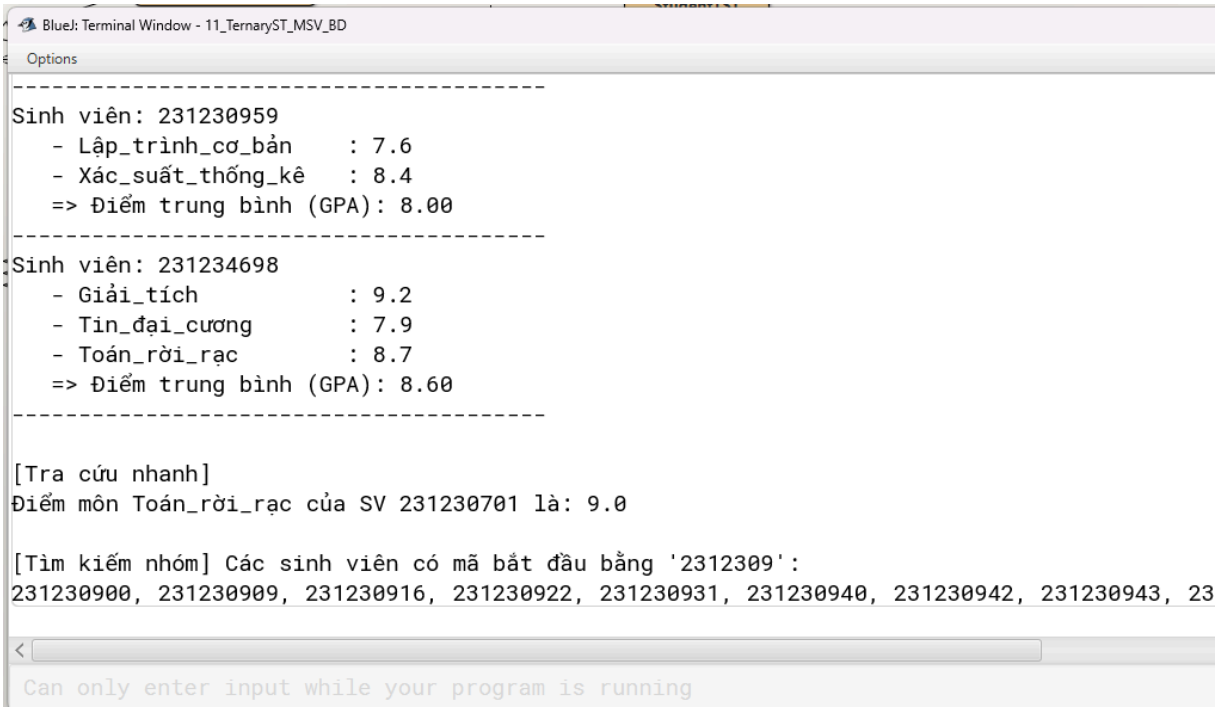
Code:

StudentTST.java

```
public class StudentTST extends VietnameseTST<VietnameseTST<Double>> {
    public void putStudent(String studentId, String subject, Double grade) {
        VietnameseTST<Double> transcript = get(studentId);
        if (transcript == null) {
            transcript = new VietnameseTST<>();
            put(studentId, transcript);
        }
        transcript.put(subject, grade);
    }
    public Double getGrade(String studentId, String subject) {
        VietnameseTST<Double> transcript = get(studentId);

        if (transcript == null) return null;
```

```
        return transcript.get(subject);
    }
    public Iterable<String> getSubjects(String studentId) {
        VietnameseTST<Double> transcript = get(studentId);
        if (transcript == null) return new Queue<String>();
        return transcript.keys();
    }
}
```

Kết quả:



Bài tập 12: Quản lý điểm sinh viên bằng B-Tree bậc 4

Code:

BTree.java

```
public class BTree<Key extends Comparable<Key>, Value>  {
    private static final int M = 4;

    private Node root;
    private int height;
    private int n;

    private static final class Node {
        private int m;
        private Entry[] children = new Entry[M];

        private Node(int k) {
            m = k;
        }
    }

    private static class Entry {
```

```
        private Comparable key;
        private Object val;
        private Node next;

        public Entry(Comparable key, Object val, Node next) {
            this.key  = key;
            this.val  = val;
            this.next = next;
        }
    }

    public BTree() {
        root = new Node(0);
    }

    public int size() { return n; }
    public boolean isEmpty() { return size() == 0; }
    public int height() { return height; }

    public Value get(Key key) {
        if (key == null) throw new IllegalArgumentException("Key không được
null");
        return search(root, key, height);
    }

    private Value search(Node x, Key key, int ht) {
        Entry[] children = x.children;

        if (ht == 0) {
            for (int j = 0; j < x.m; j++) {
                if (eq(key, children[j].key)) return (Value) children[j].val;
            }
        }
        else {
            for (int j = 0; j < x.m; j++) {
                if (j+1 == x.m || less(key, children[j+1].key))
                    return search(children[j].next, key, ht-1);
            }
        }
        return null;
    }

    public void put(Key key, Value val) {
        if (key == null) throw new IllegalArgumentException("Key không được
null");

        Node u = insert(root, key, val, height);
        n++;

        if (u == null) return;

        Node t = new Node(2);
        t.children[0] = new Entry(root.children[0].key, null, root);
```

```
        t.children[1] = new Entry(u.children[0].key, null, u);
        root = t;
        height++;
    }

    private Node insert(Node h, Key key, Value val, int ht) {
        int j;
        Entry t = new Entry(key, val, null);

        if (ht == 0) {
            for (j = 0; j < h.m; j++) {
                if (less(key, h.children[j].key)) break;
            }
        }
        else {
            for (j = 0; j < h.m; j++) {
                if ((j+1 == h.m) || less(key, h.children[j+1].key)) {
                    Node u = insert(h.children[j++].next, key, val, ht-1);
                    if (u == null) return null;

                    t.key = u.children[0].key;
                    t.val = null;
                    t.next = u;
                    break;
                }
            }
        }

        for (int i = h.m; i > j; i--)
            h.children[i] = h.children[i-1];
        h.children[j] = t;
        h.m++;

        if (h.m < M) return null;
        else         return split(h);
    }

    private Node split(Node h) {
        Node t = new Node(M/2);
        h.m = M/2;
        for (int j = 0; j < M/2; j++)
            t.children[j] = h.children[M/2+j];
        return t;
    }

    public String toString() {
        return toString(root, height, "") + "\n";
    }

    private String toString(Node h, int ht, String indent) {
        StringBuilder s = new StringBuilder();
        Entry[] children = h.children;
```

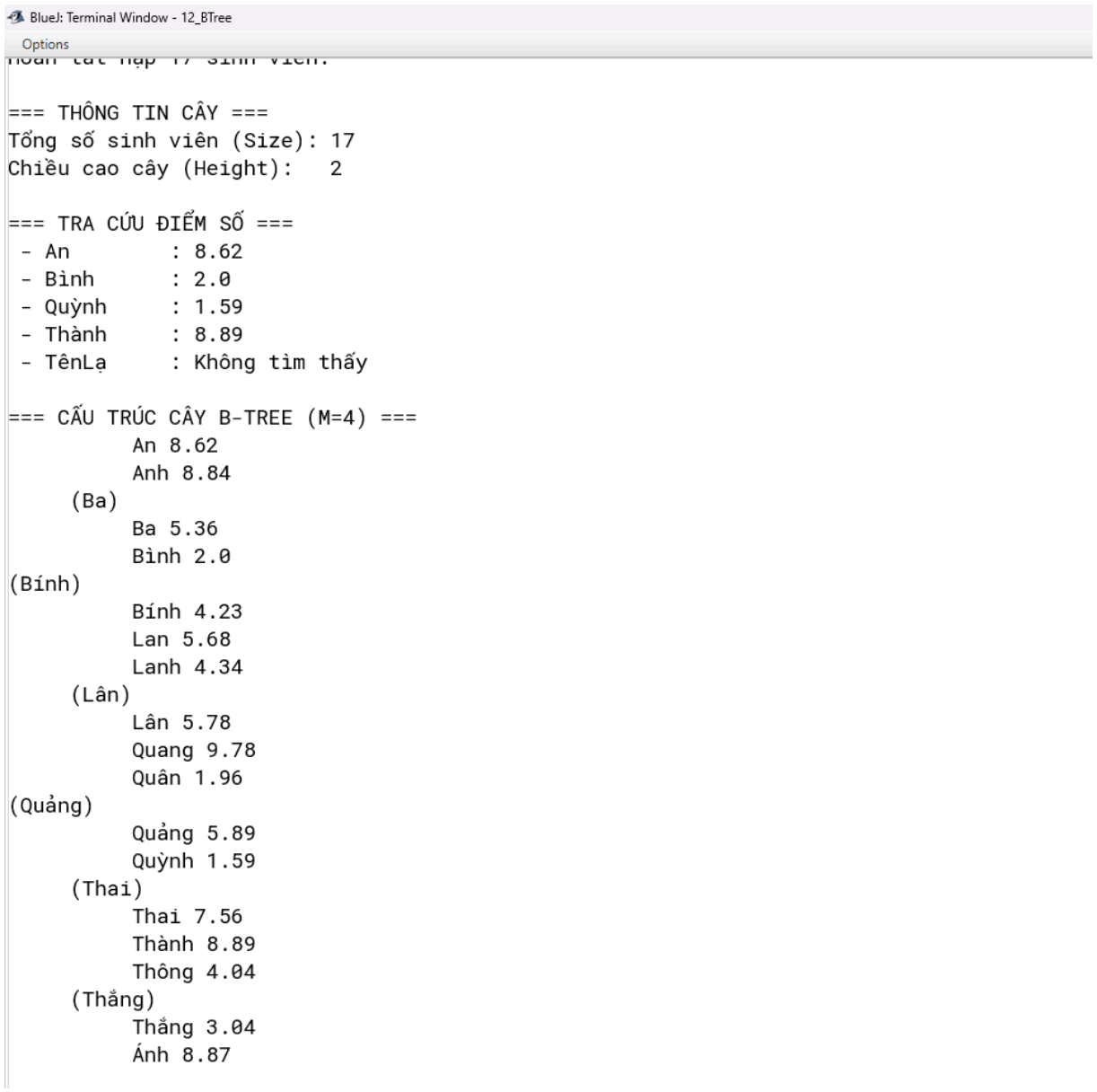
```
        if (ht == 0) {
            for (int j = 0; j < h.m; j++) {
                s.append(indent + children[j].key + " " + children[j].val +
"\n");
            }
        }
        else {
            for (int j = 0; j < h.m; j++) {
                if (j > 0) s.append(indent + "(" + children[j].key + ")\n");
                s.append(toString(children[j].next, ht-1, indent + "     "));
            }
        }
        return s.toString();
    }

    private boolean less(Comparable k1, Comparable k2) {
        return k1.compareTo(k2) < 0;
    }
    private boolean eq(Comparable k1, Comparable k2) {
        return k1.compareTo(k2) == 0;
    }
}
```

VietnameseBTree.java

```
import java.util.Random;

public class VietnameseBTree extends BTree<String, Double> {

    private static final String[] NAMES = {
        "An", "Anh", "Ánh", "Ba", "Bình", "Bính", "Lan", "Lân", "Lanh",
        "Quang", "Quảng", "Quỳnh", "Quân", "Thai", "Thành", "Thắng", "Thông"
    };

    private static final Random RANDOM = new Random();

    public VietnameseBTree() {
        super();
        initialize();
    }
    private void initialize() {
        System.out.println("Đang nạp dữ liệu vào B-Tree (M=4)...");
        for (String name : NAMES) {
            double score = Math.round(RANDOM.nextDouble() * 10.0 * 100.0) /
100.0;

            put(name, score);
        }
        System.out.println("Hoàn tất nạp " + NAMES.length + " sinh viên.\n");
    }

    public static void main(String[] args) {
        VietnameseBTree tree = new VietnameseBTree();
```

```
        System.out.println("=== THÔNG TIN CÂY ===");
        System.out.println("Tổng số sinh viên (Size): " + tree.size());
        System.out.println("Chiều cao cây (Height):   " + tree.height());
        System.out.println();

        System.out.println("=== TRA CỨU ĐIỂM SỐ ===");
        String[] testNames = {"An", "Bình", "Quỳnh", "Thành", "TênLạ"};
        for (String name : testNames) {
            Double score = tree.get(name);
            System.out.printf(" - %-10s: %s\n", name, (score != null ? score :
"Không tìm thấy"));
        }
        System.out.println();

        System.out.println("=== CẤU TRÚC CÂY B-TREE (M=4) ===");
        System.out.println(tree);
    }
}
```

Kết quả:



Bài tập 13: Tìm kiếm chuỗi mẫu Tiếng Việt bằng thuật toán KMP

Code:

KMP.java

```
public class KMP {
    private final int R;
    private final int m;
    private int[][] dfa;

    public KMP(String pat) {
        this.R = 256;
        this.m = pat.length();

        dfa = new int[R][m];
        dfa[pat.charAt(0)][0] = 1;

        for (int x = 0, j = 1; j < m; j++) {
```

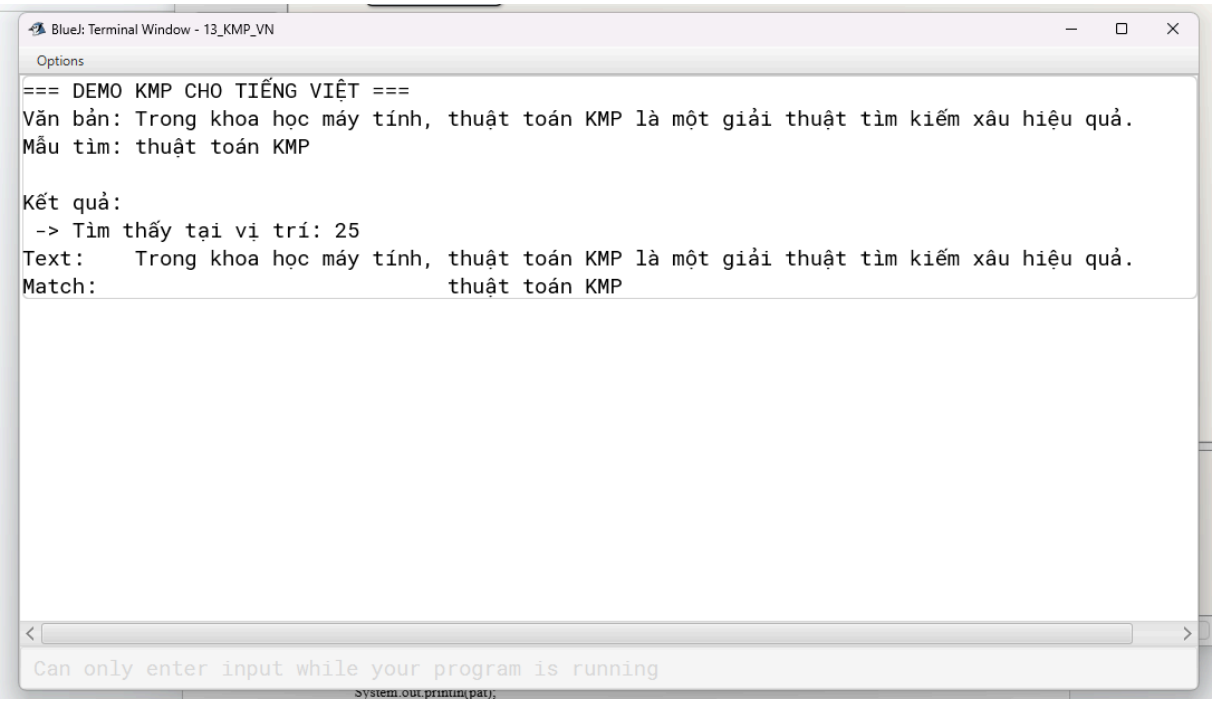
```
            for (int c = 0; c < R; c++)
                dfa[c][j] = dfa[c][x];

            dfa[pat.charAt(j)][j] = j+1;

            x = dfa[pat.charAt(j)][x];
        }
    }

    public KMP(char[] pattern, int R) {
        this.R = R;
        this.m = pattern.length;

        dfa = new int[R][m];
        dfa[pattern[0]][0] = 1;
        for (int x = 0, j = 1; j < m; j++) {
            for (int c = 0; c < R; c++)
                dfa[c][j] = dfa[c][x];
            dfa[pattern[j]][j] = j+1;
            x = dfa[pattern[j]][x];
        }
    }

    public int search(String txt) {
        int n = txt.length();
        int i, j;
        for (i = 0, j = 0; i < n && j < m; i++) {
            j = dfa[txt.charAt(i)][j];
        }
        if (j == m) return i - m;
        return n;
    }

    public int search(char[] text) {
        int n = text.length;
        int i, j;
        for (i = 0, j = 0; i < n && j < m; i++) {
            j = dfa[text[i]][j];
        }
        if (j == m) return i - m;
        return n;
    }
}
```

KMPVN.java

```
public class KMPVN extends KMP {
    private final Alphabet alphabet;

    public KMPVN(String pat) {
        this(pat, VietnameseAlphabet.VIETNAMESE_ALPHABET);
    }
```

```
    public KMPVN(String pat, Alphabet alphabet) {

        super(toIndexedCharArray(pat, alphabet), alphabet.radix());
        this.alphabet = alphabet;
    }

    private static char[] toIndexedCharArray(String s, Alphabet alphabet) {
        char[] indexed = new char[s.length()];
        for (int i = 0; i < s.length(); i++) {

            indexed[i] = (char) alphabet.toIndex(s.charAt(i));
        }
        return indexed;
    }

    @Override
    public int search(String txt) {
        return super.search(toIndexedCharArray(txt, alphabet));
    }


    public static void main(String[] args) {

        String pat = "thuật toán KMP";

        String txt = "Trong khoa học máy tính, thuật toán KMP là một giải thuật
tìm kiếm xâu hiệu quả.";

        System.out.println("=== DEMO KMP CHO TIẾNG VIỆT ===");
        System.out.println("Văn bản: " + txt);
        System.out.println("Mẫu tìm: " + pat);

        KMPVN kmp = new KMPVN(pat);
        int offset = kmp.search(txt);

        System.out.println("\nKết quả:");
        if (offset < txt.length()) {
            System.out.println(" -> Tìm thấy tại vị trí: " + offset);

            System.out.println("Text:    " + txt);
            System.out.print("Match:   ");
            for (int i = 0; i < offset; i++) {
                System.out.print(" ");
            }
            System.out.println(pat);
        } else {
            System.out.println(" -> Không tìm thấy mẫu trong văn bản.");
        }
    }
}
```

Kết quả:



Bài tập 14: Tìm kiếm mẫu câu dài trong file văn bản Tiếng Việt bằng KMP.

Code:

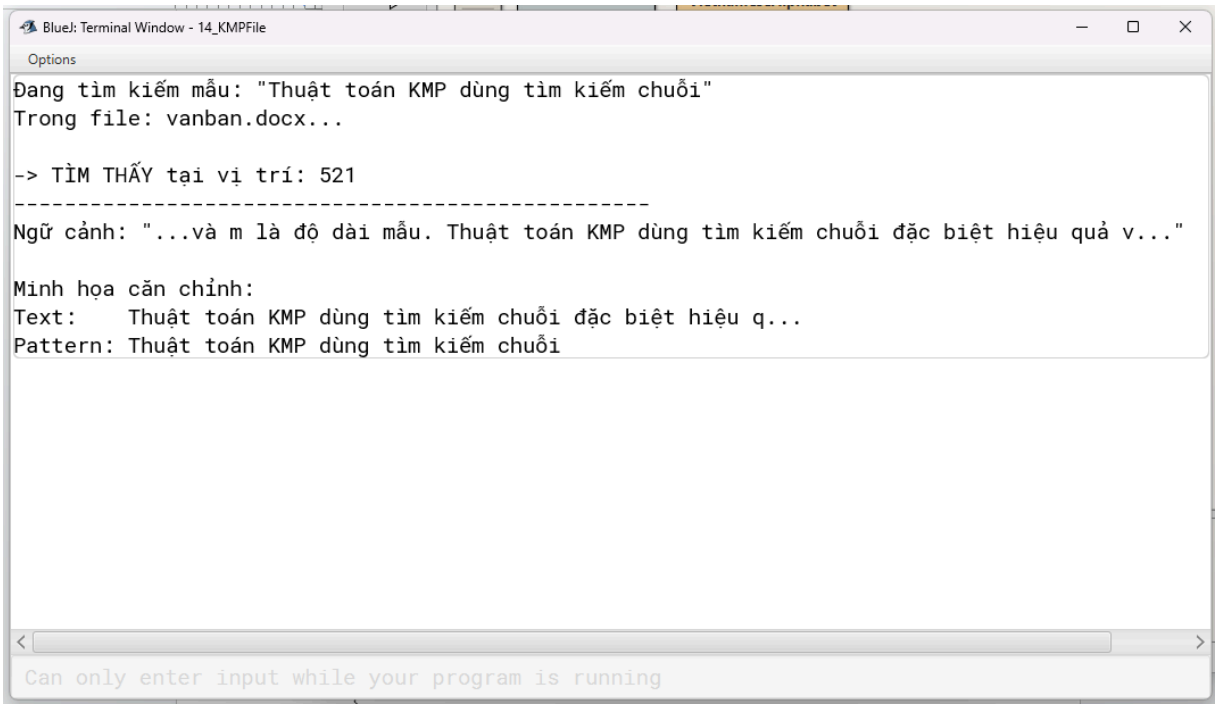
KMPFile.java

```
import java.io.*;
import org.apache.poi.xwpf.usermodel.XWPFDocument;
import org.apache.poi.xwpf.usermodel.XWPFParagraph;

public class KMPFile extends KMP {
    private final Alphabet alphabet;

    public KMPFile(String pat, Alphabet alphabet) {
        super(toIndexedCharArray(pat, alphabet), alphabet.radix());
        this.alphabet = alphabet;
    }


    private static char[] toIndexedCharArray(String s, Alphabet alphabet) {
        char[] indexed = new char[s.length()];
        for (int i = 0; i < s.length(); i++) {
            indexed[i] = (char) alphabet.toIndex(s.charAt(i));
        }
        return indexed;
    }

    private String readDocxFile(String filePath) throws IOException {
        StringBuilder textBuilder = new StringBuilder();
        try (FileInputStream fis = new FileInputStream(new File(filePath));
             XWPFDocument document = new XWPFDocument(fis)) {

            for (XWPFParagraph para : document.getParagraphs()) {
                String text = para.getText().trim();
```

```
                if (!text.isEmpty()) {
                    textBuilder.append(text).append(" ");
                }
            }
        }

        return textBuilder.toString().replaceAll("\\s+", " ");
    }


    public int searchInFile(String filePath) {
        try {
            String text = readDocxFile(filePath);

            char[] indexedText = toIndexedCharArray(text, alphabet);

            return super.search(indexedText);

        } catch (IOException e) {
            System.err.println("Lỗi đọc file .docx: " + e.getMessage());
            return -1;
        }
    }

    public static void main(String[] args) {
        String pat = "Thuật toán KMP dùng tìm kiếm chuỗi";
        String filePath = "vanban.docx";

        if (!new File(filePath).exists()) {
            System.out.println("Không tìm thấy file: " + filePath);
            return;
        }

        System.out.println("Đang tìm kiếm mẫu: \"" + pat + "\"");
        System.out.println("Trong file: " + filePath + "...");

        KMPFile kmp = new KMPFile(pat, VietnameseAlphabet.VIETNAMESE_ALPHABET);
        int offset = kmp.searchInFile(filePath);

        if (offset >= 0 && offset < Integer.MAX_VALUE) {
             try {
                String text = kmp.readDocxFile(filePath);

                if (offset > text.length()) {
                     System.out.println("-> Kết quả: Không tìm thấy mẫu trong
văn bản.");
                     return;
                }

                System.out.println("\n-> TÌM THẤY tại vị trí: " + offset);

System.out.println("--------------------------------------------------");
```

```
                int start = Math.max(0, offset - 20);
                int end = Math.min(text.length(), offset + pat.length() + 20);
                String snippet = text.substring(start, end);

                System.out.println("Ngữ cảnh: \"..." + snippet + "...\"");

                System.out.println("\nMinh họa căn chỉnh:");
                String displayTxt = text.substring(offset,
Math.min(text.length(), offset + 50));
                System.out.println("Text:    " + displayTxt + "...");
                System.out.println("Pattern: " + pat);

            } catch (IOException e) {
                e.printStackTrace();
            }
        } else {
            System.out.println("-> Kết quả: Không tìm thấy mẫu hoặc có lỗi xảy
ra.");
        }
    }
}
```

Kết quả:



Bài tập 15: Tìm kiếm chuỗi Tiếng Việt sử dụng thuật toán Boyer-Moore

Code:

BoyerMoore.java

```
public class BoyerMoore {
    protected int R;
    protected int[] right;
```

```
    private char[] pattern;
    private String pat;

    public BoyerMoore(String pat) {
        this.R = 256;
        this.pat = pat;

        right = new int[R];
        for (int c = 0; c < R; c++)
            right[c] = -1;

        for (int j = 0; j < pat.length(); j++)
            right[pat.charAt(j)] = j;
    }

    public BoyerMoore(char[] pattern, int R) {
        this.R = R;
        this.pattern = new char[pattern.length];
        for (int j = 0; j < pattern.length; j++)
            this.pattern[j] = pattern[j];

        right = new int[R];
        for (int c = 0; c < R; c++)
            right[c] = -1;
        for (int j = 0; j < pattern.length; j++)
            right[pattern[j]] = j;
    }

    public int search(String txt) {
        int m = pat.length();
        int n = txt.length();
        int skip;

        for (int i = 0; i <= n - m; i += skip) {
            skip = 0;
            for (int j = m-1; j >= 0; j--) {
                if (pat.charAt(j) != txt.charAt(i+j)) {

                    skip = Math.max(1, j - right[txt.charAt(i+j)]);
                    break;
                }
            }
            if (skip == 0) return i;
        }
        return n;
    }

    public int search(char[] text) {
        int m = pattern.length;
        int n = text.length;
        int skip;
        for (int i = 0; i <= n - m; i += skip) {
            skip = 0;
```

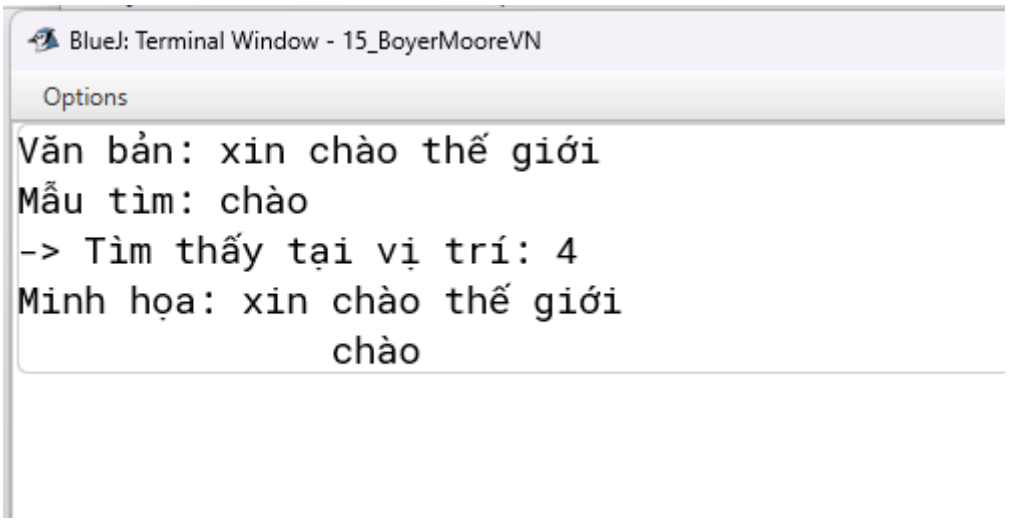
```
            for (int j = m-1; j >= 0; j--) {
                if (pattern[j] != text[i+j]) {
                    skip = Math.max(1, j - right[text[i+j]]);
                    break;
                }
            }
            if (skip == 0) return i;
        }
        return n;
    }
}
```

VietnameseBoyerMoore.java

```
public class VietnameseBoyerMoore extends BoyerMoore {
    private final Alphabet alphabet;
    private int[] right;
    private char[] pattern;
    private String pat;


    public VietnameseBoyerMoore(String pat) {
        super(new char[0], 0);
        this.alphabet = VietnameseAlphabet.VIETNAMESE_ALPHABET;
        this.R = alphabet.radix();
        this.pat = pat;
        this.pattern = pat.toCharArray();
        initialize();
    }

    private void initialize() {
        right = new int[R];
        for (int c = 0; c < R; c++) {
            right[c] = -1;
        }

        for (int j = 0; j < pat.length(); j++) {
            char c = pat.charAt(j);
            if (alphabet.contains(c)) {
                right[alphabet.toIndex(c)] = j;
            }
        }
    }

    @Override
    public int search(String txt) {
        int m = pat.length();
        int n = txt.length();
        int skip;

        for (int i = 0; i <= n - m; i += skip) {
            skip = 0;
            for (int j = m - 1; j >= 0; j--) {
```

```
                char textChar = txt.charAt(i + j);

                if (pat.charAt(j) != textChar) {
                    if (alphabet.contains(textChar)) {

                        skip = Math.max(1, j -
right[alphabet.toIndex(textChar)]);
                    } else {
                        skip = j + 1;
                    }
                    break;
                }
            }
            if (skip == 0) return i;
        }
        return n;
    }

    public static void main(String[] args) {
        String pat = args.length > 0 ? args[0] : "chào";
        String txt = args.length > 1 ? args[1] : "xin chào thế giới";

        Alphabet alphabet = VietnameseAlphabet.VIETNAMESE_ALPHABET;
        for (char c : pat.toCharArray()) {
            if (!alphabet.contains(c)) {
                System.out.println("Lỗi: Mẫu chứa ký tự không được hỗ trợ: " +
c);
                return;
            }
        }

        System.out.println("Văn bản: " + txt);
        System.out.println("Mẫu tìm: " + pat);

        VietnameseBoyerMoore bm = new VietnameseBoyerMoore(pat);
        int offset = bm.search(txt);

        if (offset < txt.length()) {
            System.out.println("-> Tìm thấy tại vị trí: " + offset);
            System.out.print("Minh họa: ");
            System.out.println(txt);
            System.out.print("          ");
            for (int i = 0; i < offset; i++) System.out.print(" ");
            System.out.println(pat);
        } else {
            System.out.println("-> Không tìm thấy.");
        }
    }
}
```

Kết quả:



Bài tập 16: Tìm kiếm xâu con Tiếng Việt bằng thuật toán Rabin-Karp

Code:

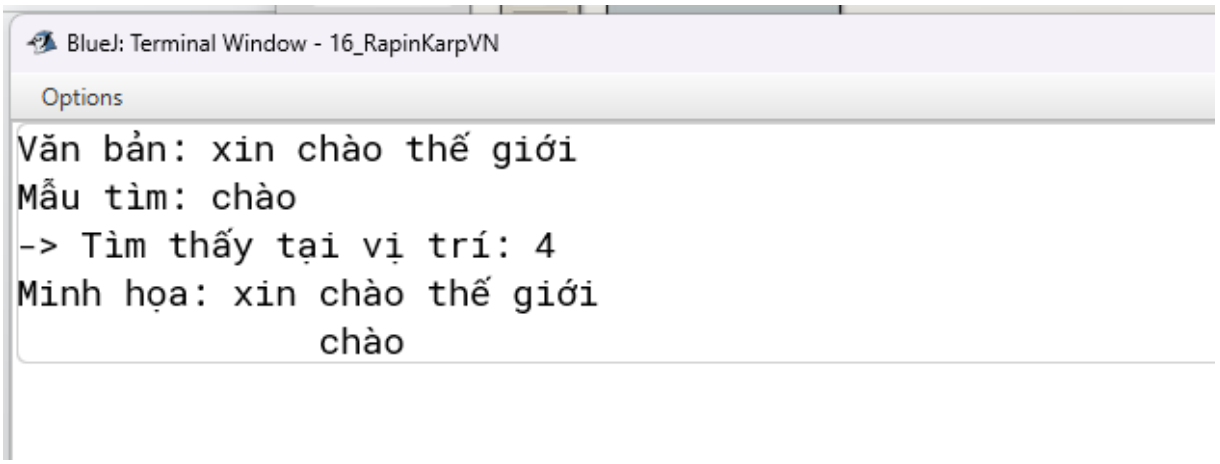
VietnameseRabinKarp.java

```
import java.math.BigInteger;
import java.util.Random;

public class VietnameseRabinKarp extends RabinKarp {
    private final Alphabet alphabet; // Bảng chữ cái Tiếng Việt
    private String pat;              // Chuỗi mẫu
    private long patHash;            // Giá trị Hash của mẫu
    private int m;                   // Độ dài mẫu
    private long q;                  // Số nguyên tố lớn (để chia lấy dư)
    private int R;                   // Cơ số (Kích thước bảng chữ cái)
    private long RM;                 // R^(M-1) % q

    public VietnameseRabinKarp(String pat) {
        super("");

        this.alphabet = VietnameseAlphabet.VIETNAMESE_ALPHABET;
        this.R = alphabet.radix();
        this.pat = pat;
        this.m = pat.length();
        this.q = longRandomPrime();

        for (char c : pat.toCharArray()) {
            if (!alphabet.contains(c)) {
                throw new IllegalArgumentException("Mẫu chứa ký tự không hỗ
trợ: " + c);
            }
        }
        RM = 1;
        for (int i = 1; i <= m - 1; i++) {
            RM = (R * RM) % q;
        }

        patHash = hash(pat, m);
```

```
    }

    private long hash(String key, int m) {
        long h = 0;
        for (int j = 0; j < m; j++) {
            char c = key.charAt(j);

            h = (R * h + alphabet.toIndex(c)) % q;
        }
        return h;
    }
    private boolean check(String txt, int i) {
        for (int j = 0; j < m; j++) {
            if (pat.charAt(j) != txt.charAt(i + j)) {
                return false;
            }
        }
        return true;
    }

    @Override
    public int search(String txt) {
        int n = txt.length();
        if (n < m) return n;

        long txtHash = 0;
        for (int j = 0; j < m; j++) {
            char c = txt.charAt(j);
            int index = alphabet.contains(c) ? alphabet.toIndex(c) : R - 1;
            txtHash = (R * txtHash + index) % q;
        }
        if ((patHash == txtHash) && check(txt, 0)) {
            return 0;
        }

        for (int i = m; i < n; i++) {
            char leadChar = txt.charAt(i - m);
            int leadIndex = alphabet.contains(leadChar) ?
alphabet.toIndex(leadChar) : R - 1;

            txtHash = (txtHash + q - (RM * leadIndex) % q) % q;

            char trailChar = txt.charAt(i);
            int trailIndex = alphabet.contains(trailChar) ?
alphabet.toIndex(trailChar) : R - 1;

            txtHash = (txtHash * R + trailIndex) % q;

            int offset = i - m + 1;
            if ((patHash == txtHash) && check(txt, offset)) {
                return offset;
            }
        }
```

```
        return n;
    }

    private static long longRandomPrime() {
        BigInteger prime = BigInteger.probablePrime(31, new Random());
        return prime.longValue();
    }

    public static void main(String[] args) {
        String pat = "chào";
        String txt = "xin chào thế giới";

        if (args.length >= 2) {
            pat = args[0];
            txt = args[1];
        }

        System.out.println("Văn bản: " + txt);
        System.out.println("Mẫu tìm: " + pat);

        try {
            VietnameseRabinKarp searcher = new VietnameseRabinKarp(pat);
            int offset = searcher.search(txt);

            // In kết quả
            if (offset < txt.length()) {
                System.out.println("-> Tìm thấy tại vị trí: " + offset);
                System.out.print("Minh họa: ");
                System.out.println(txt);
                System.out.print("          ");
                for (int i = 0; i < offset; i++) {
                    System.out.print(" ");
                }
                System.out.println(pat);
            } else {
                System.out.println("-> Không tìm thấy.");
            }
        } catch (IllegalArgumentException e) {
            System.err.println("Lỗi: " + e.getMessage());
        }
    }
}
```

Kết quả:



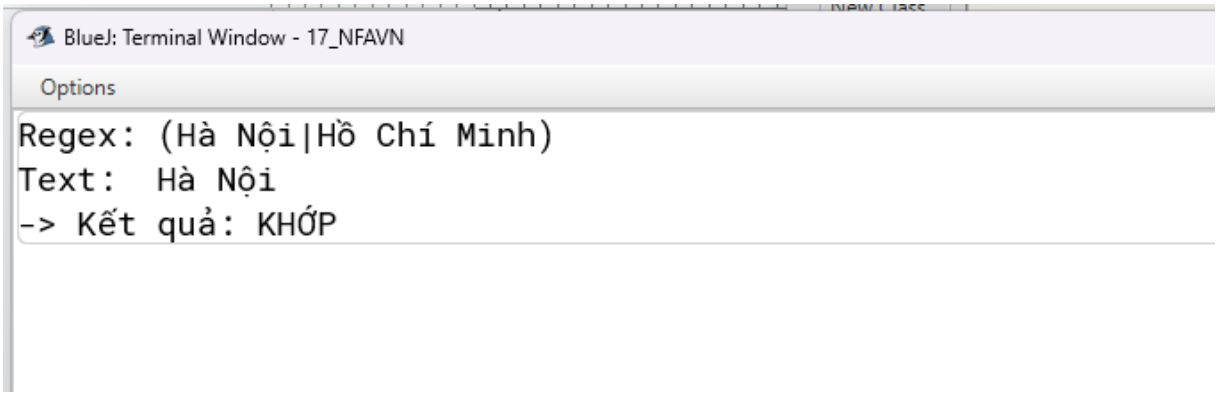
Bài tập 17: Kiểm tra khớp chuỗi Tiếng Việt bằng NFA

Code:
NFAVN.java

```
public class NFAVN extends NFA {

    private final VietnameseAlphabet alphabet;

    public NFAVN(String regexp) {
        super(convertToIndices(regexp));
        this.alphabet = VietnameseAlphabet.VIETNAMESE_ALPHABET;
    }

    private static String convertToIndices(String s) {
        VietnameseAlphabet alpha = VietnameseAlphabet.VIETNAMESE_ALPHABET;
        StringBuilder sb = new StringBuilder();

        for (int i = 0; i < s.length(); i++) {
            char c = s.charAt(i);

            if (c == '(' || c == ')' || c == '*' || c == '|') {
                sb.append(c);
            }

            else if (alpha.contains(c)) {
                int index = alpha.toIndex(c);
                sb.append((char) index);
            } else {
                throw new IllegalArgumentException("Ký tự không hỗ trợ: " + c);
            }
        }
        return sb.toString();
    }

    @Override
    public boolean recognizes(String txt) {
        for (int i = 0; i < txt.length(); i++) {
            char c = txt.charAt(i);
```

```
            if (!alphabet.contains(c)) {
                throw new IllegalArgumentException("Văn bản chứa ký tự lạ: " +
c);
            }
        }

        String convertedTxt = convertToIndices(txt);

        return super.recognizes(convertedTxt);
    }

    public static void main(String[] args) {
        String regexp = "(Hà Nội|Hồ Chí Minh)";
        String txt = "Hà Nội";

        if (args.length >= 2) {
            regexp = "(" + args[0] + ")";
            txt = args[1];
        }

        System.out.println("Regex: " + regexp);
        System.out.println("Text:  " + txt);

        try {
            NFAVN nfa = new NFAVN(regexp);
            boolean result = nfa.recognizes(txt);

            System.out.println("-> Kết quả: " + (result ? "KHỚP" : "KHÔNG
KHỚP"));

        } catch (IllegalArgumentException e) {
            System.err.println("Lỗi: " + e.getMessage());
        }
    }
}
```

Kết quả:



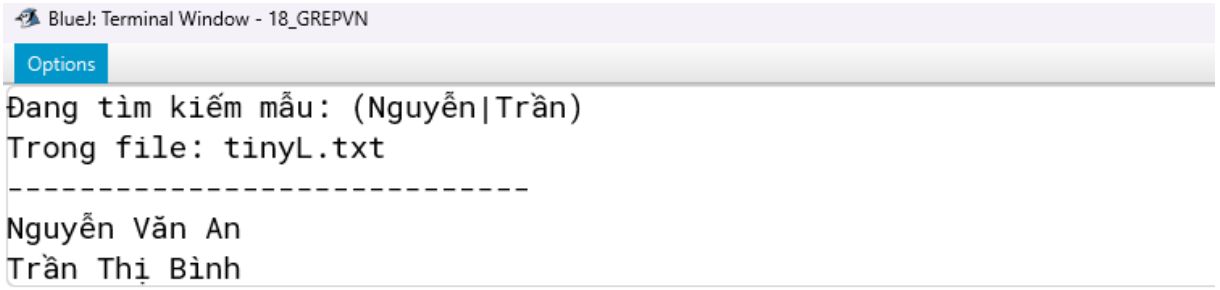
Bài tập 18: Tìm kiếm tên sinh viên trong file văn bản bằng công cụ GREP hỗ trợ Tiếng Việt

Code:

GREPVN.java

```
import java.io.File;

public class GREPVN {

    private GREPVN() { }

    public static void main(String[] args) {
        // 1. Kiểm tra tham số đầu vào
        if (args.length < 2) {
            System.err.println("Cách dùng: java GREPVN <regex> <filename>");
            System.err.println("Ví dụ: java GREPVN \"(Nguyễn|Trần)\"
tinyL.txt");
            return;
        }

        String pattern = args[0];
        String fileName = args[1];

        String regexp = "(.*" + pattern + ".*)";

        try {
            NFAVN nfa = new NFAVN(regexp);

            In in = new In(fileName);

            System.out.println("Đang tìm kiếm mẫu: " + pattern);
            System.out.println("Trong file: " + fileName);
            System.out.println("------------------------------");

            while (in.hasNextLine()) {
                String line = in.readLine();
                try {
                    if (nfa.recognizes(line)) {
                        System.out.println(line);
                    }
                } catch (IllegalArgumentException e) {
                }
            }

        } catch (IllegalArgumentException e) {
            System.err.println("Lỗi Regex hoặc File: " + e.getMessage());
        } catch (Exception e) {
            System.err.println("Không thể đọc file: " + fileName);
        }
    }
}
```

Kết quả:



Bài tập 19 + 20 + 21: Nén Run-Length và Ghi ra File | Giải nén và Ghi ra File so sánh | So sánh file gốc và file sau giải nén (BinaryDump)

Code:
RunLength.java

```java
import java.io.*;

public class RunLength {
    private static final int R    = 256; // Số lượng giá trị đếm tối đa (2^8)
    private static final int LG_R = 8;   // Số bit dùng để mã hóa độ dài chạy

    private RunLength() { }

    public static void compress(String inputFilename, String outputFilename) {
        BinaryIn in = new BinaryIn(inputFilename);
        BinaryOut out = new BinaryOut(outputFilename);

        char run = 0;
        boolean old = false;
        int totalBits = 0;

        System.out.println("   [LOG] Đang nén...");

        while (!in.isEmpty()) {
            boolean b = in.readBoolean();
            totalBits++;

            if (b != old) {
                out.write(run, LG_R);
                run = 1;
                old = !old;
            } else {
                if (run == R - 1) {
                    out.write(run, LG_R);
                    run = 0;
                    out.write(run, LG_R);
                }
                run++;
            }
        }
        out.write(run, LG_R);
```

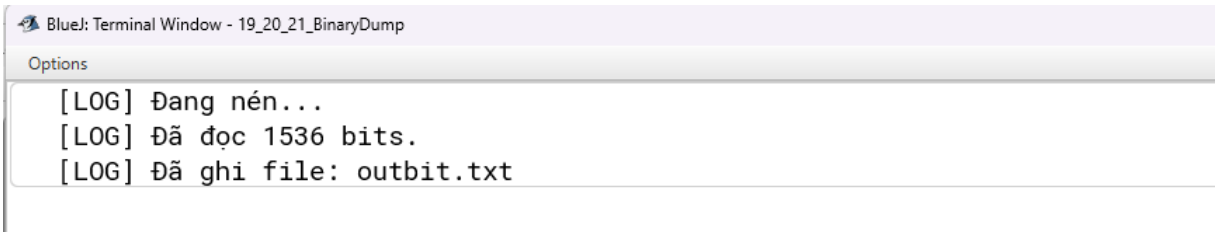
```
        out.close();

        System.out.println("   [LOG] Đã đọc " + totalBits + " bits.");
        System.out.println("   [LOG] Đã ghi file: " + outputFilename);
    }
    public static void expand(String inputFilename, String outputFilename) {
        BinaryIn in = new BinaryIn(inputFilename);
        BinaryOut out = new BinaryOut(outputFilename);

        boolean b = false;
        System.out.println("   [LOG] Đang giải nén...");

        while (!in.isEmpty()) {
            // Đọc 8 bit độ dài
            int run = in.readInt(LG_R);

            // Ghi bit b lặp lại run lần
            for (int i = 0; i < run; i++) {
                out.write(b);
            }
            b = !b; // Đảo bit
        }
        out.close();
        System.out.println("   [LOG] Giải nén hoàn tất: " + outputFilename);
    }

    public static void main(String[] args) {
        if (args.length != 3) {
            System.out.println("Usage: java RunLength [-/+] inputFile
outputFile");
            return;
        }

        String mode = args[0];
        String inputFile = args[1];
        String outputFile = args[2];

        // Kiểm tra file đầu vào có tồn tại không
        File file = new File(inputFile);
        if (!file.exists()) {
            System.out.println("Lỗi: Không tìm thấy file " + inputFile);
            return;
        }

        try {
            if (mode.equals("-")) {
                compress(inputFile, outputFile);
            } else if (mode.equals("+")) {
                expand(inputFile, outputFile);
            } else {
                System.out.println("Lỗi: Chế độ phải là '-' (nén) hoặc '+' (giải
nén)");
            }
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

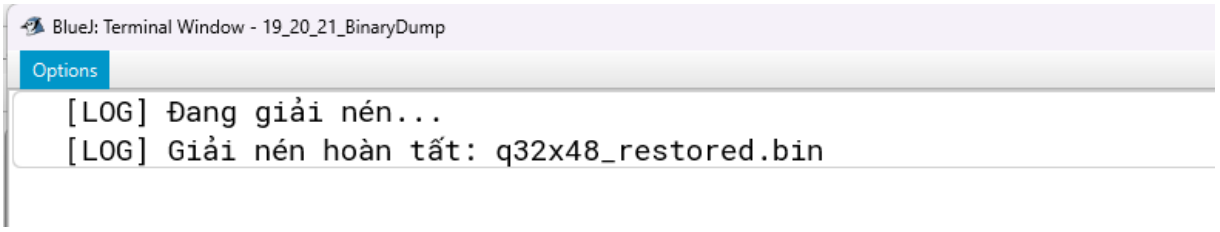
BinaryDump.java

```
import java.io.*;

public class BinaryDump {

    private BinaryDump() { }

    public static void main(String[] args) throws IOException {
        int bitsPerLine = 16; // Mặc định 16 bit/dòng nếu không chỉ định
        String filename;

        if (args.length >= 2) {
            bitsPerLine = Integer.parseInt(args[0]);
            filename = args[1];
        } else if (args.length == 1) {
            filename = args[0];
        } else {
            System.out.println("Usage: java BinaryDump <bitsPerLine>
<filename>");
            return;
        }

        File file = new File(filename);
        if (!file.exists()) {
            System.out.println("File không tồn tại: " + filename);
            return;
        }
        System.setIn(new FileInputStream(file));

        int count;
        // Đọc từng bit và in ra
        for (count = 0; !BinaryStdIn.isEmpty(); count++) {
            if (bitsPerLine != 0) {
                if (count != 0 && count % bitsPerLine == 0) StdOut.println();
            }
            if (BinaryStdIn.readBoolean()) StdOut.print(1);
            else                           StdOut.print(0);
        }

        if (bitsPerLine != 0) StdOut.println();
        System.out.println("\nTổng số bit: " + count + " bits");
    }
}
```

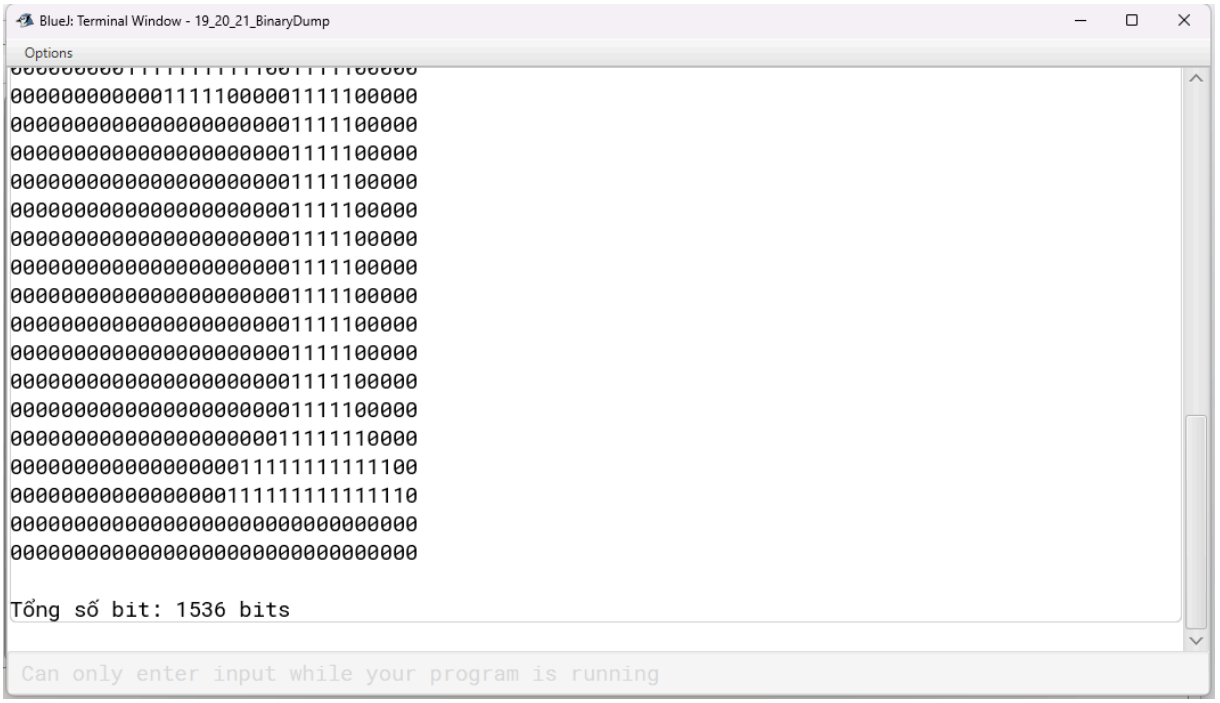
Kết quả:
Chạy RunLength.java: {"-", "q32x48.bin", "outbit.txt" }



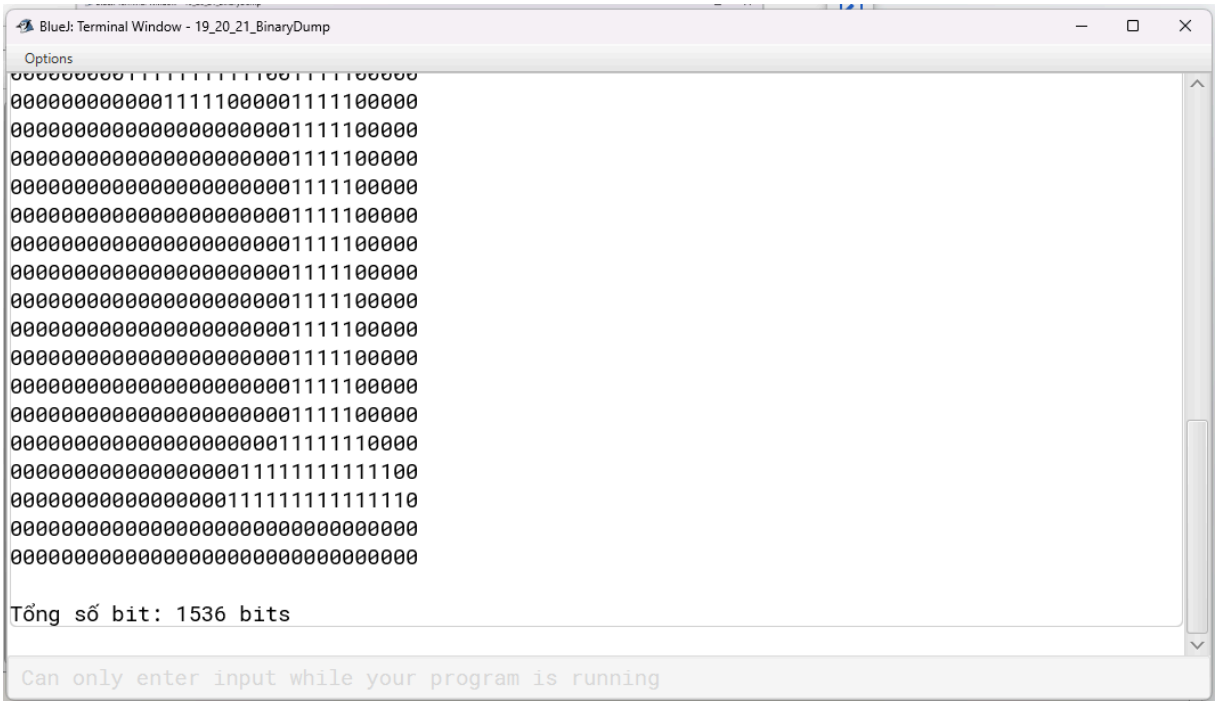
Chạy RunLength.java: {"+", "outbit.txt", "q32x48_restored.bin" }



Chạy BinaryDump.java so sánh 2 kết quả chạy:
{"32", "q32x48.bin"}



{"32", "q32x48_restored.bin"}



Bài tập 22: Nén và Giải nén danh sách sinh viên Tiếng Việt bằng mã hóa Huffman

Code:
HuffmanDSVN.java

```
import java.io.*;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.HashMap;
import java.util.Map;

public class HuffmanVietnamese extends Huffman {

    public static void compressWithAlphabet(String inputFile, String outputFile)
throws IOException {
        System.out.println("=== QUY TRÌNH NÉN ===");
        System.out.println("1. Đọc file: " + inputFile);

        byte[] bytes = Files.readAllBytes(Paths.get(inputFile));
        String s = new String(bytes, StandardCharsets.UTF_8);
        char[] input = s.toCharArray();

        System.out.println("   ✓ " + input.length + " ký tự (" + bytes.length +
" bytes)");

        System.out.println("2. Khởi tạo Vietnamese Alphabet...");
        VietnameseAlphabet alphabet = new VietnameseAlphabet();
        int R = alphabet.radix();
        System.out.println("   ✓ Alphabet size: " + R + " ký tự");

        System.out.println("3. Thống kê tần suất...");
        int[] freq = new int[R];
```

```
        // Đếm tần suất dựa trên index trong alphabet
        for (char c : input) {
            try {
                int idx = alphabet.toIndex(c);
                freq[idx]++;
            } catch (IllegalArgumentException e) {
                System.err.println("   ⚠ Ký tự không trong alphabet: '" + c +
"' (U+" +
                                   String.format("%04X", (int)c) + ")");
            }
        }

        int uniqueChars = 0;
        for (int f : freq) if (f > 0) uniqueChars++;
        System.out.println("   ✓ " + uniqueChars + " ký tự duy nhất trong
alphabet");

        System.out.println("4. Xây dựng cây Huffman...");
        Node root = buildTrie(freq);
        System.out.println("   ✓ OK");

        System.out.println("5. Tạo bảng mã...");
        String[] codeTable = new String[R];
        buildCode(codeTable, root, "");
        System.out.println("   ✓ OK");

        System.out.println("6. Ghi file nén...");
        try (BitOutputStream out = new BitOutputStream(
                new BufferedOutputStream(new FileOutputStream(outputFile)))) {

            out.writeInt(R); // Kích thước alphabet

            // Ghi cây Huffman
            writeTrie(root, out);

            // Ghi số ký tự gốc
            out.writeInt(input.length);

            // Ghi dữ liệu nén
            for (int i = 0; i < input.length; i++) {
                int idx = alphabet.toIndex(input[i]);
                String code = codeTable[idx];

                for (int j = 0; j < code.length(); j++) {
                    out.writeBit(code.charAt(j) == '1');
                }

                if ((i + 1) % 50 == 0 || i == input.length - 1) {
                    System.out.print("\r   - " + (i + 1) + "/" + input.length);
                }
            }
            System.out.println();
```

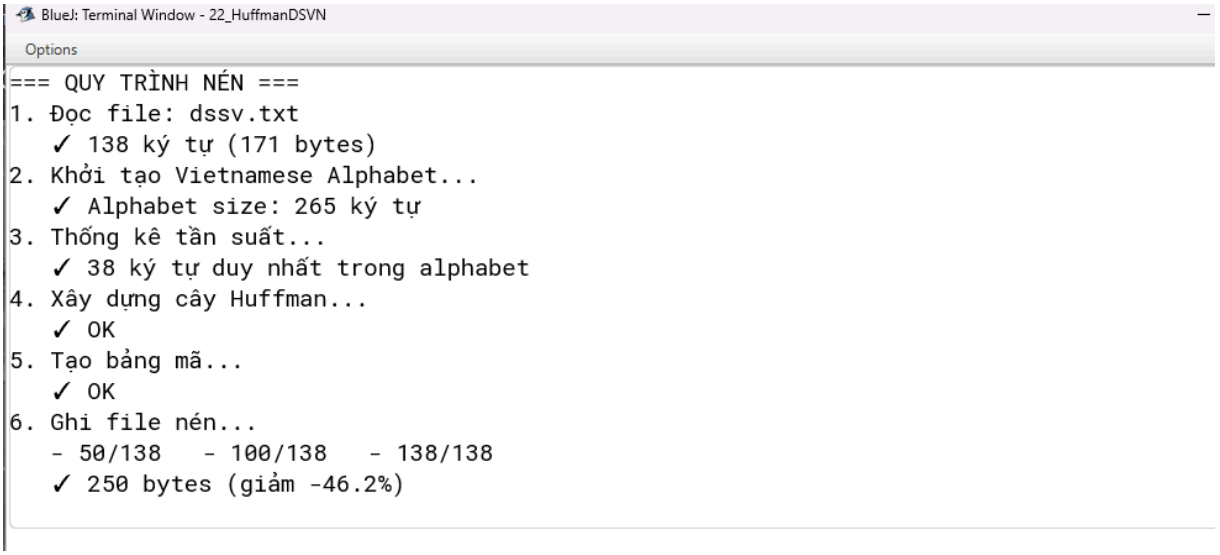
```
        }

        File outFile = new File(outputFile);
        double ratio = (1.0 - (double)outFile.length() / bytes.length) * 100;
        System.out.println("   ✓ " + outFile.length() + " bytes (giảm " +
                           String.format("%.1f%%", ratio) + ")");
        System.out.println();
    }

    public static void expandWithAlphabet(String inputFile, String outputFile)
throws IOException {
        System.out.println("=== QUY TRÌNH GIẢI NÉN ===");
        System.out.println("1. Đọc file: " + inputFile);

        try (BitInputStream in = new BitInputStream(
                new BufferedInputStream(new FileInputStream(inputFile)))) {

            System.out.println("2. Đọc thông tin alphabet...");
            int R = in.readInt();
            System.out.println("   ✓ Alphabet size: " + R);

            VietnameseAlphabet alphabet = new VietnameseAlphabet();
            if (alphabet.radix() != R) {
                System.err.println("   ⚠ Cảnh báo: Alphabet size không khớp!");
            }

            System.out.println("3. Đọc cây Huffman...");
            Node root = readTrie(in);
            System.out.println("   ✓ OK");

            System.out.println("4. Giải mã...");
            int length = in.readInt();
            System.out.println("   - " + length + " ký tự");

            StringBuilder sb = new StringBuilder(length);
            for (int i = 0; i < length; i++) {
                Node x = root;

                while (!x.isLeaf()) {
                    boolean bit = in.readBit();
                    x = bit ? x.right : x.left;
                }

                char c = alphabet.toChar(x.ch);
                sb.append(c);

                if ((i + 1) % 50 == 0 || i == length - 1) {
                    System.out.print("\r   - " + (i + 1) + "/" + length);
                }
            }
            System.out.println();

            System.out.println("5. Ghi file...");
```

```
            Files.write(Paths.get(outputFile),
sb.toString().getBytes(StandardCharsets.UTF_8));
        }

        File outFile = new File(outputFile);
        System.out.println("   ✓ " + outFile.length() + " bytes");
        System.out.println();
    }

    protected static void buildCode(String[] st, Node x, String s) {
        if (x == null) return;
        if (x.isLeaf()) {
            st[x.ch] = s.length() == 0 ? "0" : s;
        } else {
            buildCode(st, x.left, s + '0');
            buildCode(st, x.right, s + '1');
        }
    }

    private static void writeTrie(Node x, BitOutputStream out) throws
IOException {
        if (x.isLeaf()) {
            out.writeBit(true);
            out.writeInt(x.ch); // Ghi index trong alphabet
        } else {
            out.writeBit(false);
            writeTrie(x.left, out);
            writeTrie(x.right, out);
        }
    }

    private static Node readTrie(BitInputStream in) throws IOException {
        boolean isLeaf = in.readBit();
        if (isLeaf) {
            int ch = in.readInt();
            return new Node((char)ch, -1, null, null);
        }
        return new Node('\0', -1, readTrie(in), readTrie(in));
    }

    private static class BitOutputStream implements AutoCloseable {
        private OutputStream out;
        private int buffer, n;

        public BitOutputStream(OutputStream out) {
            this.out = out;
            this.buffer = 0;
            this.n = 0;
        }

        public void writeBit(boolean bit) throws IOException {
            buffer <<= 1;
            if (bit) buffer |= 1;
```

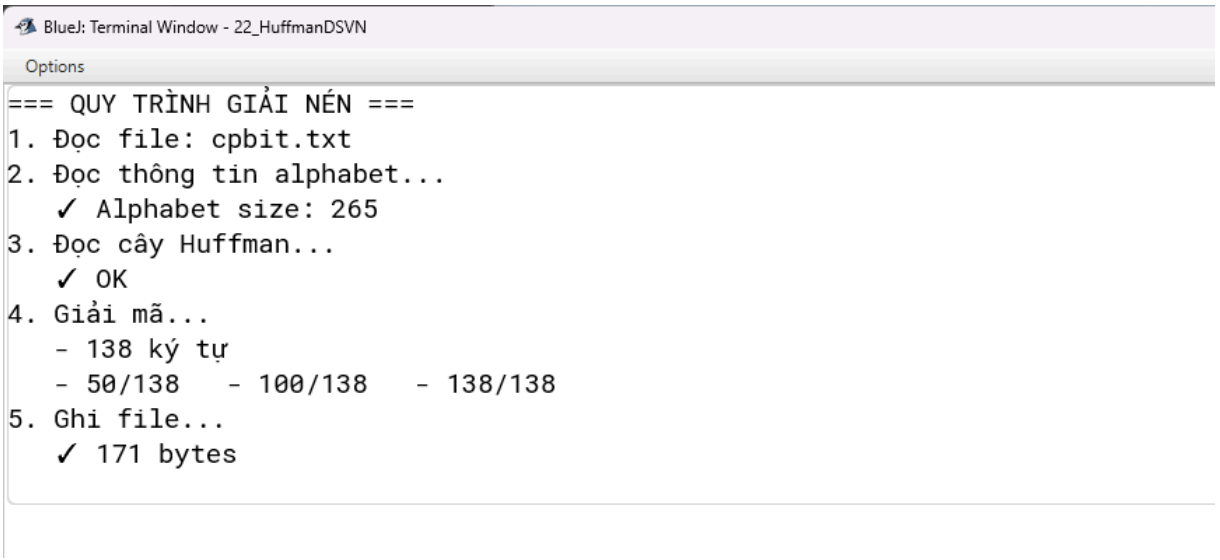
```
            n++;
            if (n == 8) {
                out.write(buffer);
                buffer = 0;
                n = 0;
            }
        }

        public void writeInt(int x) throws IOException {
            for (int i = 0; i < 32; i++) {
                writeBit(((x >>> (31 - i)) & 1) == 1);
            }
        }

        public void close() throws IOException {
            if (n > 0) {
                buffer <<= (8 - n);
                out.write(buffer);
            }
            out.flush();
            out.close();
        }
    }

    private static class BitInputStream implements AutoCloseable {
        private InputStream in;
        private int buffer, n;

        public BitInputStream(InputStream in) {
            this.in = in;
            this.buffer = 0;
            this.n = 0;
        }

        public boolean readBit() throws IOException {
            if (n == 0) {
                buffer = in.read();
                if (buffer == -1) throw new EOFException("EOF");
                n = 8;
            }
            n--;
            return ((buffer >> n) & 1) == 1;
        }

        public int readInt() throws IOException {
            int x = 0;
            for (int i = 0; i < 32; i++) {
                x = (x << 1) | (readBit() ? 1 : 0);
            }
            return x;
        }

        public void close() throws IOException {
```

```
            in.close();
        }
    }

    public static void main(String[] args) {
        if (args.length < 2) {
            System.out.println("Sử dụng:");
            System.out.println("  java HuffmanVietnamese - <input> <output>");
            System.out.println("  java HuffmanVietnamese + <input> <output>");
            return;
        }

        try {
            String mode = args[0];
            String input = args[1];
            String output = args.length > 2 ? args[2] :
                            (mode.equals("-") ? input + ".bin" : "output.txt");

            if (mode.equals("-")) {
                compressWithAlphabet(input, output);
            } else if (mode.equals("+")) {
                expandWithAlphabet(input, output);
            } else {
                System.out.println("Lỗi: mode phải là '-' hoặc '+'");
            }
        } catch (Exception e) {
            System.err.println("LỖI: " + e.getMessage());
            e.printStackTrace();
        }
    }
}
```

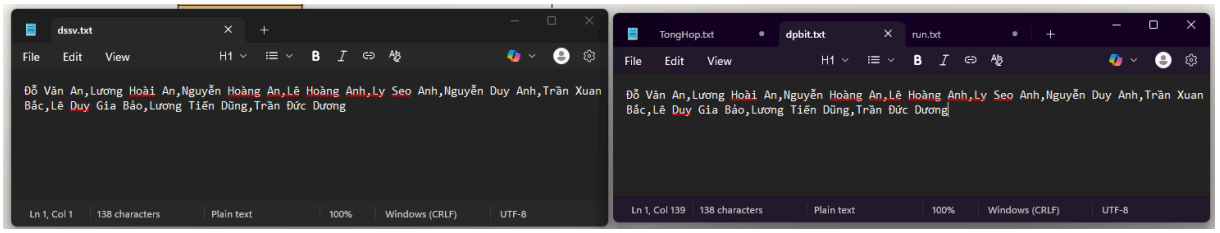
Kết quả:
Nén: {"-", "dssv.txt", "cpbit.txt" }



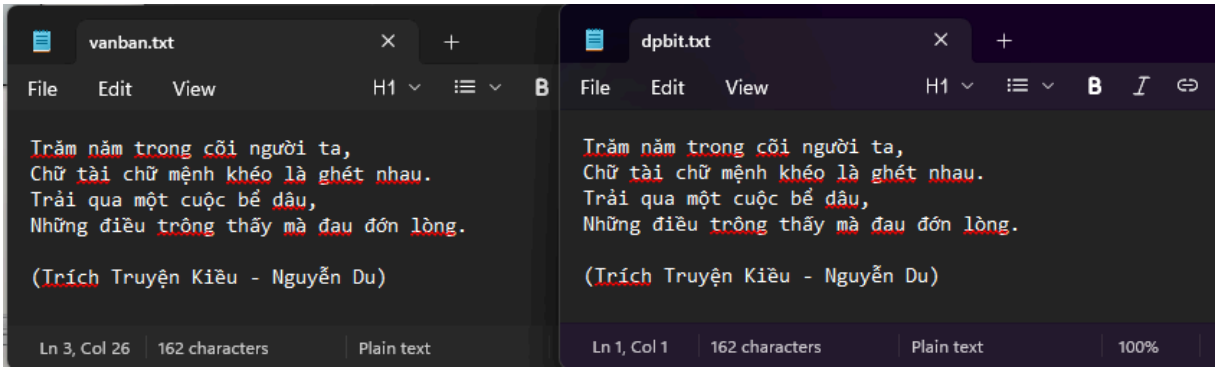
Giải nén: {"+", "cpbit.txt", "dpbit.txt" }



So sánh: Trùng Khớp



Bài tập 23: Nén và Giải nén một đoạn văn bản Tiếng Việt bất kỳ.
Kết quả: Sau khi nén và giải nén, văn bản trùng khớp.



Bài tập 24: Nén và Giải nén LZW cho Tiếng Việt.

Code:

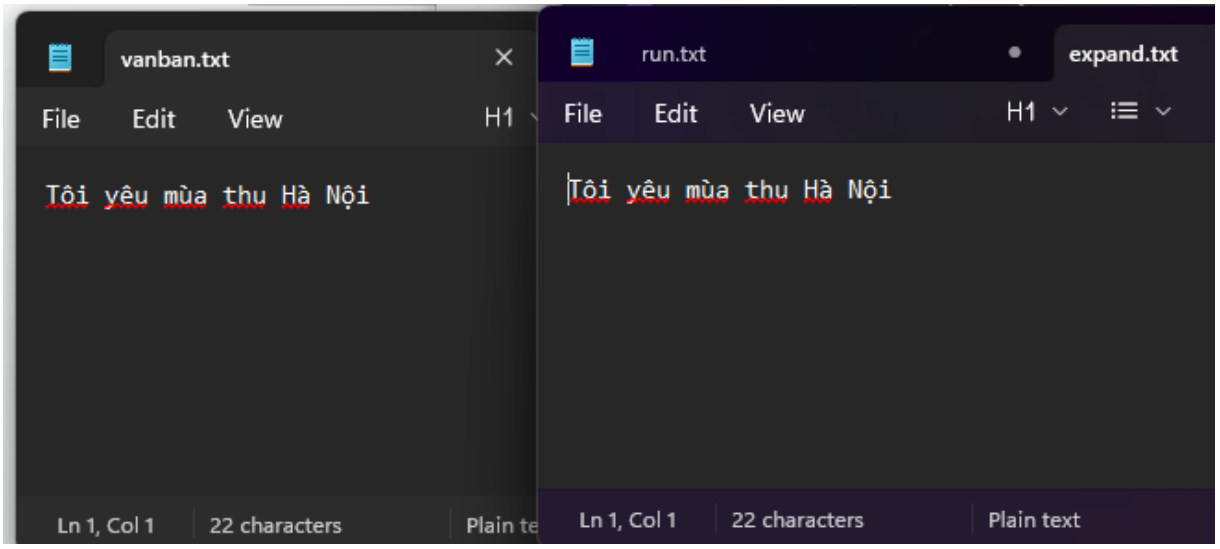
```
import java.io.*;
import java.nio.file.*;
import java.util.HashSet;

public class LZWVN {
    private static final VietnameseAlphabet ALPHABET =
VietnameseAlphabet.VIETNAMESE_ALPHABET;
    private static final int R = ALPHABET.radix();  // Kích thước bảng chữ cái
    private static final int L = 65536;             // Số lượng codewords = 2^16
    private static final int W = 16;                // Độ rộng codeword (16 bit)

    // Không instantiate
    private LZWVN() { }
```

```
    public static void compress(String inputFile, String outputFile) {
        try {
            // Đọc file input với UTF-8
            String input = new String(Files.readAllBytes(Paths.get(inputFile)),
"UTF-8");

            // Kiểm tra input có ký tự ngoài bảng chữ cái không
            HashSet<Character> alphabetSet = new HashSet<>();
            for (int i = 0; i < ALPHABET.radix(); i++) {
                alphabetSet.add(ALPHABET.toChar(i));
            }
            for (char c : input.toCharArray()) {
                if (!alphabetSet.contains(c)) {
                    throw new IllegalArgumentException("Input contains character
not in VietnameseAlphabet: " + c);
                }
            }

            BinaryOut bout = new BinaryOut(outputFile);
            TST<Integer> st = new TST<Integer>();

            // Khởi tạo symbol table với các ký tự từ VietnameseAlphabet
            for (int i = 0; i < R; i++) {
                st.put("" + ALPHABET.toChar(i), i);
            }

            int code = R + 1;  // Codeword cho EOF

            while (input.length() > 0) {
                String s = st.longestPrefixOf(input);  // Tìm prefix dài nhất
                bout.write(st.get(s), W);              // Ghi codeword
                int t = s.length();
                if (t < input.length() && code < L) {  // Thêm codeword mới
                    st.put(input.substring(0, t + 1), code++);
                }
                input = input.substring(t);
            }
            bout.write(R, W);  // Ghi EOF
            bout.close();
        } catch (IOException e) {
            throw new RuntimeException("Error during compression: " +
e.getMessage());
        }
    }

    public static void expand(String inputFile, String outputFile) {
        try {
            BinaryIn bin = new BinaryIn(inputFile);
            // Ghi output với UTF-8
            BufferedWriter writer =
Files.newBufferedWriter(Paths.get(outputFile),
java.nio.charset.StandardCharsets.UTF_8);
```

```
            String[] st = new String[L];
            int i; // Next available codeword value

            // Khởi tạo symbol table
            for (i = 0; i < R; i++) {
                st[i] = "" + ALPHABET.toChar(i);
            }
            st[i++] = "";  // Lookahead cho EOF

            int codeword = bin.readInt(W);
            if (codeword == R) return;  // Empty input
            String val = st[codeword];

            while (true) {
                writer.write(val);  // Ghi string ra file
                codeword = bin.readInt(W);
                if (codeword == R) break;
                String s = st[codeword];
                if (i == codeword) s = val + val.charAt(0); // Special case hack
                if (i < L) st[i++] = val + s.charAt(0);
                val = s;
            }
            writer.close();
        } catch (IOException e) {
            throw new RuntimeException("Error during decompression: " +
e.getMessage());
        }
    }

    public static void main(String[] args) {
        if (args.length != 3) {
            throw new IllegalArgumentException("Usage: java LZWVN [-/+] [input
file] [output file]");
        }

        String mode = args[0];
        String inputFile = args[1];
        String outputFile = args[2];

        // Kiểm tra file input
        File file = new File(inputFile);
        if (!file.exists() || !file.isFile()) {
            throw new IllegalArgumentException("Input file does not exist: " +
inputFile);
        }

        try {
            if (mode.equals("-")) {
                compress(inputFile, outputFile);
                System.out.println("Compression completed: " + outputFile);
            } else if (mode.equals("+")) {
                expand(inputFile, outputFile);
```

```
                System.out.println("Decompression completed: " + outputFile);
            } else {
                throw new IllegalArgumentException("Illegal command line
argument: must be '-' or '+'");
            }
        } catch (Exception e) {
            System.err.println("Error processing files: " + e.getMessage());
        }
    }
}
```

Kết quả: Sau khi nén và giải nén, văn bản trùng khớp.



Bài tập 25: Cài đặt các phương thức tìm kiếm phạm vi cho BST
Mở rộng lớp BST (dựa trên tài liệu Algorithms 4th Edition) để thêm chức năng truy vấn phạm vi.

Phương thức size(lo, hi): Trả về số lượng key thỏa mãn lo <= key <= hi.
Phương thức search(lo, hi): Trả về danh sách (Queue<Key>) các key thỏa mãn điều kiện trên theo thứ tự tăng dần.

Code:
BSTExtended.java

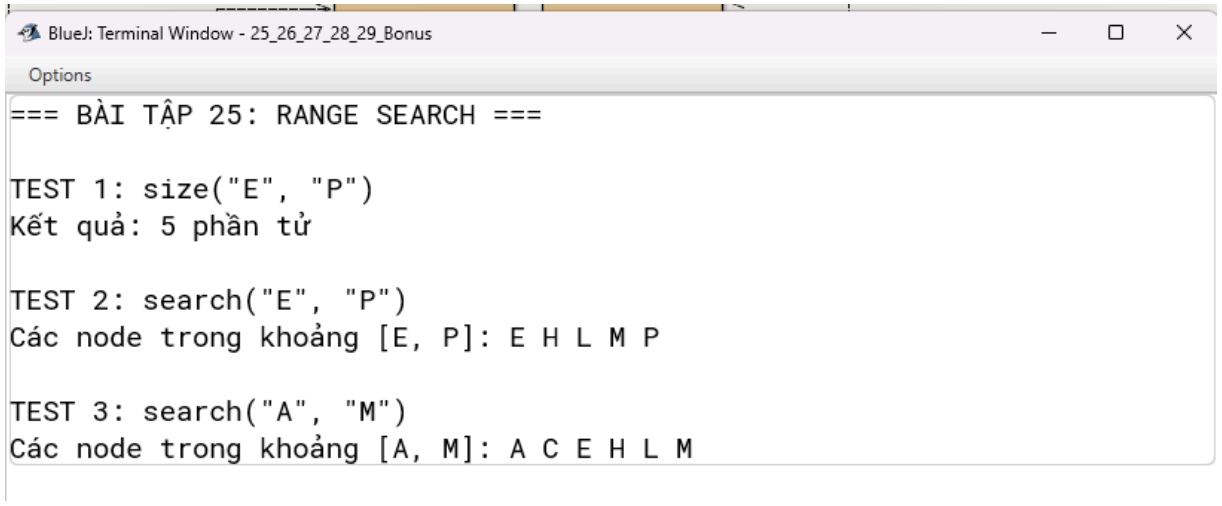
```
import java.util.NoSuchElementException;

public class BSTExtended<Key extends Comparable<Key>, Value> {
    private Node root;

    private class Node {
        private Key key;
        private Value val;
        private Node left, right;
        private int size;

        public Node(Key key, Value val, int size) {
            this.key = key;
            this.val = val;
```

```
            this.size = size;
        }
    }

    public BSTExtended() {
    }

    public boolean isEmpty() {
        return size() == 0;
    }

    public int size() {
        return size(root);
    }

    private int size(Node x) {
        if (x == null) return 0;
        else return x.size;
    }

    public void put(Key key, Value val) {
        if (key == null) throw new IllegalArgumentException("calls put() with a
null key");
        root = put(root, key, val);
    }

    private Node put(Node x, Key key, Value val) {
        if (x == null) return new Node(key, val, 1);
        int cmp = key.compareTo(x.key);
        if      (cmp < 0) x.left  = put(x.left,  key, val);
        else if (cmp > 0) x.right = put(x.right, key, val);
        else              x.val   = val;
        x.size = 1 + size(x.left) + size(x.right);
        return x;
    }

    public Value get(Key key) {
        return get(root, key);
    }

    private Value get(Node x, Key key) {
        if (key == null) throw new IllegalArgumentException("calls get() with a
null key");
        if (x == null) return null;
        int cmp = key.compareTo(x.key);
        if      (cmp < 0) return get(x.left, key);
        else if (cmp > 0) return get(x.right, key);
        else              return x.val;
    }
    public int size(Key lo, Key hi) {
        if (lo == null || hi == null)
            throw new IllegalArgumentException("arguments cannot be null");
```

```
        Queue<Node> queue = search(root, lo, hi);
        return queue.size();
    }

    public Queue<Node> search(Key lo, Key hi) {
        if (lo == null || hi == null)
            throw new IllegalArgumentException("arguments cannot be null");

        return search(root, lo, hi);
    }

    private Queue<Node> search(Node x, Key lo, Key hi) {
        Queue<Node> queue = new Queue<Node>();

        if (x == null) return queue;

        int cmplo = lo.compareTo(x.key);
        int cmphi = hi.compareTo(x.key);

        // 1. Đệ quy cây con trái (nếu lo < x.key)
        if (cmplo < 0) {
            Queue<Node> leftQueue = search(x.left, lo, hi);
            // Chuyển tất cả node từ leftQueue vào queue
            while (!leftQueue.isEmpty()) {
                queue.enqueue(leftQueue.dequeue());
            }
        }

        // 2. Kiểm tra node gốc (nếu lo <= x.key <= hi)
        if (cmplo <= 0 && cmphi >= 0) {
            queue.enqueue(x);
        }

        // 3. Đệ quy cây con phải (nếu hi > x.key)
        if (cmphi > 0) {
            Queue<Node> rightQueue = search(x.right, lo, hi);
            // Chuyển tất cả node từ rightQueue vào queue
            while (!rightQueue.isEmpty()) {
                queue.enqueue(rightQueue.dequeue());
            }
        }

        return queue;
    }

    public static void main(String[] args) {
        BSTExtended<String, Integer> bst = new BSTExtended<>();

        // Tạo cây BST
        String[] keys = {"S", "E", "A", "R", "C", "H", "X", "M", "P", "L"};
        for (int i = 0; i < keys.length; i++) {
            bst.put(keys[i], i);
        }
```

```
        System.out.println("=== BÀI TẬP 25: RANGE SEARCH ===\n");

        // Test 1: size(lo, hi)
        String lo = "E";
        String hi = "P";
        System.out.println("TEST 1: size(\"" + lo + "\", \"" + hi + "\")");
        System.out.println("Kết quả: " + bst.size(lo, hi) + " phần tử\n");

        // Test 2: search(lo, hi)
        System.out.println("TEST 2: search(\"" + lo + "\", \"" + hi + "\")");
        Queue<BSTExtended<String, Integer>.Node> result = bst.search(lo, hi);

        System.out.print("Các node trong khoảng [" + lo + ", " + hi + "]: ");
        for (BSTExtended<String, Integer>.Node node : result) {
            System.out.print(node.key + " ");
        }
        System.out.println("\n");

        // Test 3: Khoảng khác
        lo = "A";
        hi = "M";
        System.out.println("TEST 3: search(\"" + lo + "\", \"" + hi + "\")");
        result = bst.search(lo, hi);

        System.out.print("Các node trong khoảng [" + lo + ", " + hi + "]: ");
        for (BSTExtended<String, Integer>.Node node : result) {
            System.out.print(node.key + " ");
        }
        System.out.println();
    }
}
```

Kết quả:



Bài tập 26: Cho n đoạn ngang và m đoạn dọc. Tìm các điểm giao cắt giữa đoạn ngang và đoạn dọc.

Code:

LineIntersection.java

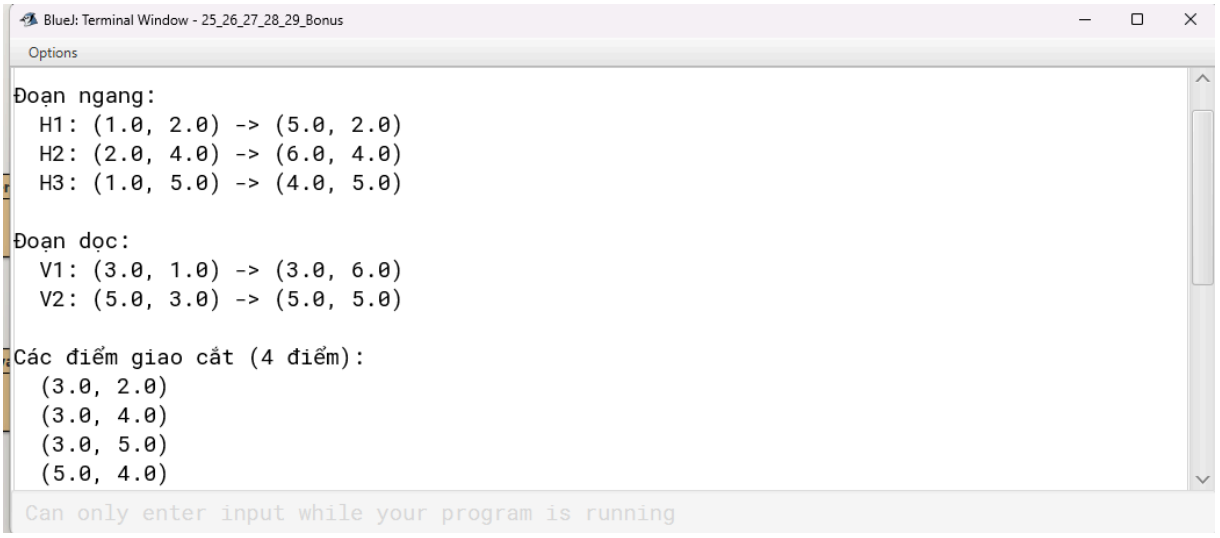
```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class LineIntersection {

    static class Segment {
        Point2D p1, p2;  // 2 điểm đầu cuối
        boolean isHorizontal;  // true nếu là đoạn ngang

        public Segment(Point2D p1, Point2D p2) {
            this.p1 = p1;
            this.p2 = p2;
            this.isHorizontal = (p1.y() == p2.y());
        }

        public double getY() {
            return p1.y();  // y của đoạn ngang
        }

        public double getX() {
            return p1.x();  // x của đoạn dọc
        }

        public double getMinX() {
            return Math.min(p1.x(), p2.x());
        }

        public double getMaxX() {
            return Math.max(p1.x(), p2.x());
        }

        public double getMinY() {
            return Math.min(p1.y(), p2.y());
        }

        public double getMaxY() {
            return Math.max(p1.y(), p2.y());
        }
    }

    static class Event implements Comparable<Event> {
        double x;  // tọa độ x của sự kiện
        int type;  // 0: bắt đầu đoạn ngang, 1: đoạn dọc, 2: kết thúc đoạn ngang
        Segment segment;

        public Event(double x, int type, Segment segment) {
            this.x = x;
            this.type = type;
            this.segment = segment;
        }
```

```
        @Override
        public int compareTo(Event other) {
            if (this.x != other.x) {
                return Double.compare(this.x, other.x);
            }
            // Ưu tiên: bắt đầu -> dọc -> kết thúc
            return Integer.compare(this.type, other.type);
        }
    }

    public static List<Point2D> findIntersections(List<Segment> horizontal,
List<Segment> vertical) {
        List<Point2D> intersections = new ArrayList<>();
        List<Event> events = new ArrayList<>();

        for (Segment h : horizontal) {
            events.add(new Event(h.getMinX(), 0, h));  // Bắt đầu
            events.add(new Event(h.getMaxX(), 2, h));  // Kết thúc
        }

        for (Segment v : vertical) {
            events.add(new Event(v.getX(), 1, v));
        }

        Collections.sort(events);

        BSTExtended<Double, Segment> activeSegments = new BSTExtended<>();

        for (Event event : events) {
            if (event.type == 0) {
                // Bắt đầu đoạn ngang: thêm vào BST
                activeSegments.put(event.segment.getY(), event.segment);

            } else if (event.type == 2) {
                // Kết thúc đoạn ngang: xóa khỏi BST (set value = null)
                activeSegments.put(event.segment.getY(), null);

            } else {
                // Đoạn dọc: tìm tất cả đoạn ngang giao với nó
                Segment v = event.segment;
                double yMin = v.getMinY();
                double yMax = v.getMaxY();

                // Tìm tất cả đoạn ngang có y trong khoảng [yMin, yMax]
                Queue<BSTExtended<Double, Segment>.Node> nodes =
                    activeSegments.search(yMin, yMax);

                // Kiểm tra từng đoạn ngang
                for (BSTExtended<Double, Segment>.Node node : nodes) {
                    if (node.val != null) {  // Đoạn còn active
                        Segment h = node.val;
                        double x = v.getX();
                        double y = h.getY();
```

```
                        if (x >= h.getMinX() && x <= h.getMaxX()) {
                            intersections.add(new Point2D(x, y));
                        }
                    }
                }
            }
        }

        return intersections;
    }

    // Hàm main để test
    public static void main(String[] args) {
        System.out.println("=== BÀI TẬP 26: TÌM GIAO ĐIỂM ĐOẠN NGANG VÀ DỌC
===\n");

        // Tạo các đoạn ngang
        List<Segment> horizontal = new ArrayList<>();
        horizontal.add(new Segment(new Point2D(1, 2), new Point2D(5, 2)));
        horizontal.add(new Segment(new Point2D(2, 4), new Point2D(6, 4)));
        horizontal.add(new Segment(new Point2D(1, 5), new Point2D(4, 5)));

        // Tạo các đoạn dọc
        List<Segment> vertical = new ArrayList<>();
        vertical.add(new Segment(new Point2D(3, 1), new Point2D(3, 6)));
        vertical.add(new Segment(new Point2D(5, 3), new Point2D(5, 5)));

        System.out.println("Đoạn ngang:");
        for (int i = 0; i < horizontal.size(); i++) {
            Segment h = horizontal.get(i);
            System.out.println("  H" + (i+1) + ": " + h.p1 + " -> " + h.p2);
        }

        System.out.println("\nĐoạn dọc:");
        for (int i = 0; i < vertical.size(); i++) {
            Segment v = vertical.get(i);
            System.out.println("  V" + (i+1) + ": " + v.p1 + " -> " + v.p2);
        }

        // Tìm giao điểm
        List<Point2D> intersections = findIntersections(horizontal, vertical);

        System.out.println("\nCác điểm giao cắt (" + intersections.size() + "
điểm):");
        for (Point2D p : intersections) {
            System.out.println("  " + p);
        }

        System.out.println("\n=== TEST 2: Ví dụ phức tạp hơn ===\n");

        // Test case 2
        List<Segment> h2 = new ArrayList<>();
```

```
        h2.add(new Segment(new Point2D(0, 1), new Point2D(8, 1)));
        h2.add(new Segment(new Point2D(1, 3), new Point2D(7, 3)));
        h2.add(new Segment(new Point2D(2, 5), new Point2D(9, 5)));

        List<Segment> v2 = new ArrayList<>();
        v2.add(new Segment(new Point2D(2, 0), new Point2D(2, 6)));
        v2.add(new Segment(new Point2D(5, 2), new Point2D(5, 6)));
        v2.add(new Segment(new Point2D(8, 0), new Point2D(8, 4)));

        System.out.println("Đoạn ngang: " + h2.size() + " đoạn");
        System.out.println("Đoạn dọc: " + v2.size() + " đoạn");

        List<Point2D> intersections2 = findIntersections(h2, v2);

        System.out.println("\nSố điểm giao cắt: " + intersections2.size());
        System.out.println("Chi tiết:");
        for (Point2D p : intersections2) {
            System.out.println("  " + p);
        }
    }
}
```

Kết quả:



Bài tập 27: 2D-Tree. Cho n điểm Mi(xi, yi) và hình chữ nhật (a, b, c, d). Hỏi có các điểm Mi nào nằm trong hình chữ nhật. Cho 1 điểm nằm ngoài hình chữ nhật, tìm điểm gần nhất trong n điểm đã cho.

Code:

TwoDTree.java

```
public class TwoDTree {

    private Node root;
    private int size;

    private class Node {
        Point2D point;      // Điểm tại node
```

```
        Node left, right;   // Con trái, con phải
        boolean isVertical; // true: chia theo x, false: chia theo y

        public Node(Point2D point, boolean isVertical) {
            this.point = point;
            this.isVertical = isVertical;
        }
    }

    public TwoDTree() {
        root = null;
        size = 0;
    }

    public boolean isEmpty() {
        return size == 0;
    }

    public int size() {
        return size;
    }

    public void insert(Point2D p) {
        if (p == null) throw new IllegalArgumentException("Point cannot be
null");
        root = insert(root, p, true);
    }

    private Node insert(Node node, Point2D p, boolean isVertical) {
        if (node == null) {
            size++;
            return new Node(p, isVertical);
        }

        if (node.point.equals(p)) return node;

        int cmp;
        if (node.isVertical) {
            cmp = Double.compare(p.x(), node.point.x());
        } else {
            cmp = Double.compare(p.y(), node.point.y());
        }

        if (cmp < 0) {
            node.left = insert(node.left, p, !node.isVertical);
        } else {
            node.right = insert(node.right, p, !node.isVertical);
        }

        return node;
    }

    public Queue<Point2D> rangeSearch(Interval2D rect) {
```

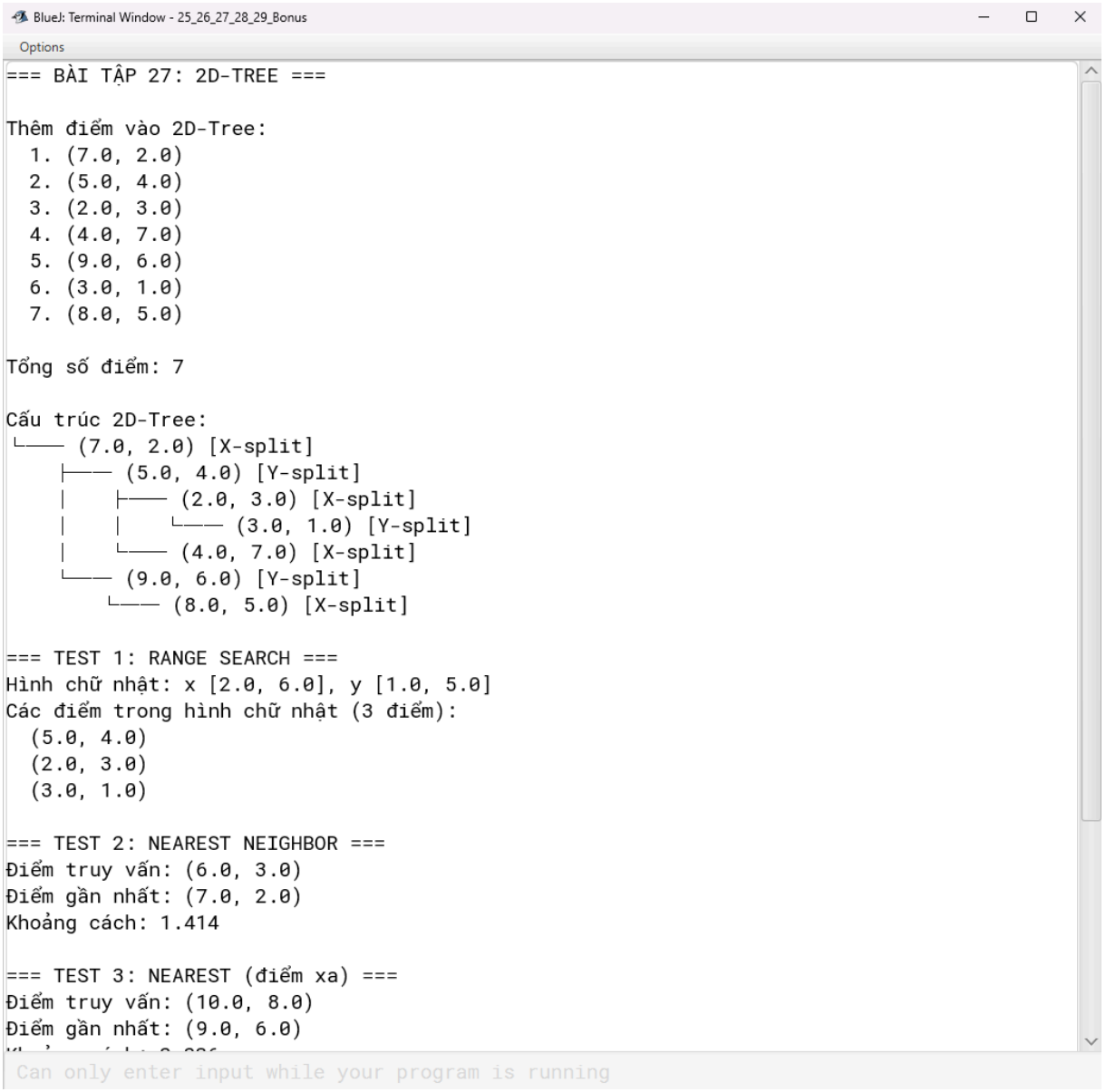
```
        if (rect == null) throw new IllegalArgumentException("Rectangle cannot
be null");

        Queue<Point2D> queue = new Queue<>();
        rangeSearch(root, rect, queue);
        return queue;
    }

    private void rangeSearch(Node node, Interval2D rect, Queue<Point2D> queue) {
        if (node == null) return;

        if (rect.contains(node.point)) {
            queue.enqueue(node.point);
        }

        if (node.isVertical) {
            if (node.left != null) {
                Interval1D xInterval = getXInterval(rect);
                if (xInterval.min() < node.point.x()) {
                    rangeSearch(node.left, rect, queue);
                }
            }
            if (node.right != null) {
                Interval1D xInterval = getXInterval(rect);
                if (xInterval.max() >= node.point.x()) {
                    rangeSearch(node.right, rect, queue);
                }
            }
        } else {
            if (node.left != null) {
                Interval1D yInterval = getYInterval(rect);
                if (yInterval.min() < node.point.y()) {
                    rangeSearch(node.left, rect, queue);
                }
            }
            if (node.right != null) {
                Interval1D yInterval = getYInterval(rect);
                if (yInterval.max() >= node.point.y()) {
                    rangeSearch(node.right, rect, queue);
                }
            }
        }
    }

    private Interval1D getXInterval(Interval2D rect) {
        String str = rect.toString();
        String xPart = str.split(" x ")[0];
        String[] bounds = xPart.replace("[", "").replace("]", "").split(", ");
        return new Interval1D(Double.parseDouble(bounds[0]),
Double.parseDouble(bounds[1]));
    }

    private Interval1D getYInterval(Interval2D rect) {
```

```
        String str = rect.toString();
        String yPart = str.split(" x ")[1];
        String[] bounds = yPart.replace("[", "").replace("]", "").split(", ");
        return new Interval1D(Double.parseDouble(bounds[0]),
Double.parseDouble(bounds[1]));
    }

    public Point2D nearest(Point2D queryPoint) {
        if (queryPoint == null) throw new IllegalArgumentException("Query point
cannot be null");
        if (isEmpty()) return null;

        return nearest(root, queryPoint, root.point);
    }

    private Point2D nearest(Node node, Point2D query, Point2D champion) {
        if (node == null) return champion;

        double champDist = query.distanceSquaredTo(champion);
        double currDist = query.distanceSquaredTo(node.point);

        if (currDist < champDist) {
            champion = node.point;
            champDist = currDist;
        }

        Node first, second;
        if (node.isVertical) {
            if (query.x() < node.point.x()) {
                first = node.left;
                second = node.right;
            } else {
                first = node.right;
                second = node.left;
            }
        } else {
            if (query.y() < node.point.y()) {
                first = node.left;
                second = node.right;
            } else {
                first = node.right;
                second = node.left;
            }
        }

        champion = nearest(first, query, champion);
        champDist = query.distanceSquaredTo(champion);

        double splitDist;
        if (node.isVertical) {
            splitDist = Math.pow(query.x() - node.point.x(), 2);
        } else {
            splitDist = Math.pow(query.y() - node.point.y(), 2);
```

```
        }

        if (splitDist < champDist) {
            champion = nearest(second, query, champion);
        }

        return champion;
    }

    public void printTree() {
        System.out.println("Cấu trúc 2D-Tree:");
        printTree(root, "", true);
    }

    private void printTree(Node node, String prefix, boolean isTail) {
        if (node == null) return;

        System.out.println(prefix + (isTail ? "└── " : "├── ") +
                           node.point + " [" + (node.isVertical ? "X-split" :
"Y-split") + "]");

        if (node.left != null || node.right != null) {
            if (node.left != null) {
                printTree(node.left, prefix + (isTail ? "    " : "│   "),
node.right == null);
            }
            if (node.right != null) {
                printTree(node.right, prefix + (isTail ? "    " : "│   "),
true);
            }
        }
    }

    public static void main(String[] args) {
        System.out.println("=== BÀI TẬP 27: 2D-TREE ===\n");

        TwoDTree tree = new TwoDTree();

        Point2D[] points = {
            new Point2D(7, 2),
            new Point2D(5, 4),
            new Point2D(2, 3),
            new Point2D(4, 7),
            new Point2D(9, 6),
            new Point2D(3, 1),
            new Point2D(8, 5)
        };

        System.out.println("Thêm điểm vào 2D-Tree:");
        for (int i = 0; i < points.length; i++) {
            tree.insert(points[i]);
            System.out.println("  " + (i+1) + ". " + points[i]);
        }
```

```
        System.out.println("\nTổng số điểm: " + tree.size());

        System.out.println();
        tree.printTree();

        // TEST 1: Range Search
        System.out.println("\n=== TEST 1: RANGE SEARCH ===");
        Interval1D xInterval = new Interval1D(2.0, 6.0);
        Interval1D yInterval = new Interval1D(1.0, 5.0);
        Interval2D rect = new Interval2D(xInterval, yInterval);

        System.out.println("Hình chữ nhật: x " + xInterval + ", y " +
yInterval);
        Queue<Point2D> inRect = tree.rangeSearch(rect);

        System.out.println("Các điểm trong hình chữ nhật (" + inRect.size() + "
điểm):");
        for (Point2D p : inRect) {
            System.out.println("  " + p);
        }

        // TEST 2: Nearest Neighbor
        System.out.println("\n=== TEST 2: NEAREST NEIGHBOR ===");
        Point2D queryPoint = new Point2D(6.0, 3.0);
        System.out.println("Điểm truy vấn: " + queryPoint);

        Point2D nearest = tree.nearest(queryPoint);
        System.out.println("Điểm gần nhất: " + nearest);
        System.out.println("Khoảng cách: " + String.format("%.3f",
queryPoint.distanceTo(nearest)));

        // TEST 3: Nearest với điểm xa
        System.out.println("\n=== TEST 3: NEAREST (điểm xa) ===");
        Point2D queryPoint2 = new Point2D(10.0, 8.0);
        System.out.println("Điểm truy vấn: " + queryPoint2);

        Point2D nearest2 = tree.nearest(queryPoint2);
        System.out.println("Điểm gần nhất: " + nearest2);
        System.out.println("Khoảng cách: " + String.format("%.3f",
queryPoint2.distanceTo(nearest2)));

        // So sánh với tất cả điểm
        System.out.println("\nKiểm tra tất cả khoảng cách:");
        for (Point2D p : points) {
            double dist = queryPoint2.distanceTo(p);
            String marker = p.equals(nearest2) ? " <-- NEAREST" : "";
            System.out.println("  " + p + ": " + String.format("%.3f", dist) +
marker);
        }
    }
}
```

Kết quả:



Bài tập 28: Cài đặt cây khoảng và phương thức iterable<value> hoặc queue<iterable1> intersect(Key lo, Key hi) và có thành phần dữ liệu BST, hàm tạo các phương thức set, get, bổ trợ.

Code:

```
import java.util.NoSuchElementException;

public class IntervalSearchTree<Value> {

    private Node root;  // Gốc của cây

    private class Node {
        Interval1D interval;  // Khoảng [lo, hi]
        Value value;          // Giá trị liên kết
        Node left, right;     // Con trái, con phải
        int size;             // Số node trong cây con
        double max;           // Giá trị max lớn nhất trong cây con

        public Node(Interval1D interval, Value value) {
```

```
            this.interval = interval;
            this.value = value;
            this.size = 1;
            this.max = interval.max();
        }
    }

    public IntervalSearchTree() {
        root = null;
    }

    public boolean isEmpty() {
        return root == null;
    }

    public int size() {
        return size(root);
    }

    private int size(Node x) {
        if (x == null) return 0;
        return x.size;
    }

    public void put(Interval1D interval, Value value) {
        if (interval == null) throw new IllegalArgumentException("Interval
cannot be null");
        if (value == null) {
            delete(interval);
            return;
        }
        root = put(root, interval, value);
    }

    private Node put(Node x, Interval1D interval, Value value) {
        if (x == null) return new Node(interval, value);

        int cmp = interval.min() < x.interval.min() ? -1 :
                  interval.min() > x.interval.min() ? 1 : 0;

        if (cmp < 0) {
            x.left = put(x.left, interval, value);
        } else if (cmp > 0) {
            x.right = put(x.right, interval, value);
        } else {
            // Cùng điểm bắt đầu, so sánh điểm kết thúc
            if (interval.max() < x.interval.max()) {
                x.left = put(x.left, interval, value);
            } else if (interval.max() > x.interval.max()) {
                x.right = put(x.right, interval, value);
            } else {
                // Khoảng trùng nhau, cập nhật value
                x.value = value;
```

```
            }
        }

        // Cập nhật size và max
        x.size = 1 + size(x.left) + size(x.right);
        x.max = max3(x.interval.max(),
                     x.left == null ? Double.NEGATIVE_INFINITY : x.left.max,
                     x.right == null ? Double.NEGATIVE_INFINITY : x.right.max);

        return x;
    }

    private double max3(double a, double b, double c) {
        return Math.max(a, Math.max(b, c));
    }

    public Value get(Interval1D interval) {
        if (interval == null) throw new IllegalArgumentException("Interval
cannot be null");
        return get(root, interval);
    }

    private Value get(Node x, Interval1D interval) {
        if (x == null) return null;

        if (x.interval.equals(interval)) return x.value;

        int cmp = interval.min() < x.interval.min() ? -1 :
                  interval.min() > x.interval.min() ? 1 : 0;

        if (cmp < 0) {
            return get(x.left, interval);
        } else if (cmp > 0) {
            return get(x.right, interval);
        } else {
            // Cùng min, so sánh max
            if (interval.max() < x.interval.max()) {
                return get(x.left, interval);
            } else if (interval.max() > x.interval.max()) {
                return get(x.right, interval);
            } else {
                return x.value;
            }
        }
    }

    public void delete(Interval1D interval) {
        if (interval == null) throw new IllegalArgumentException("Interval
cannot be null");
        root = delete(root, interval);
    }

    private Node delete(Node x, Interval1D interval) {
```

```
        if (x == null) return null;

        if (x.interval.equals(interval)) {
            // Tìm thấy node cần xóa
            if (x.left == null) return x.right;
            if (x.right == null) return x.left;

            // Node có 2 con: thay bằng min của cây con phải
            Node t = x;
            x = min(t.right);
            x.right = deleteMin(t.right);
            x.left = t.left;
        } else {
            int cmp = interval.min() < x.interval.min() ? -1 :
                      interval.min() > x.interval.min() ? 1 : 0;

            if (cmp < 0) {
                x.left = delete(x.left, interval);
            } else if (cmp > 0) {
                x.right = delete(x.right, interval);
            } else {
                if (interval.max() < x.interval.max()) {
                    x.left = delete(x.left, interval);
                } else {
                    x.right = delete(x.right, interval);
                }
            }
        }

        x.size = 1 + size(x.left) + size(x.right);
        x.max = max3(x.interval.max(),
                     x.left == null ? Double.NEGATIVE_INFINITY : x.left.max,
                     x.right == null ? Double.NEGATIVE_INFINITY : x.right.max);
        return x;
    }

    private Node min(Node x) {
        if (x.left == null) return x;
        return min(x.left);
    }

    private Node deleteMin(Node x) {
        if (x.left == null) return x.right;
        x.left = deleteMin(x.left);
        x.size = 1 + size(x.left) + size(x.right);
        x.max = max3(x.interval.max(),
                     x.left == null ? Double.NEGATIVE_INFINITY : x.left.max,
                     x.right == null ? Double.NEGATIVE_INFINITY : x.right.max);
        return x;
    }

    public Queue<Interval1D> intersect(double lo, double hi) {
        Interval1D interval = new Interval1D(lo, hi);
```

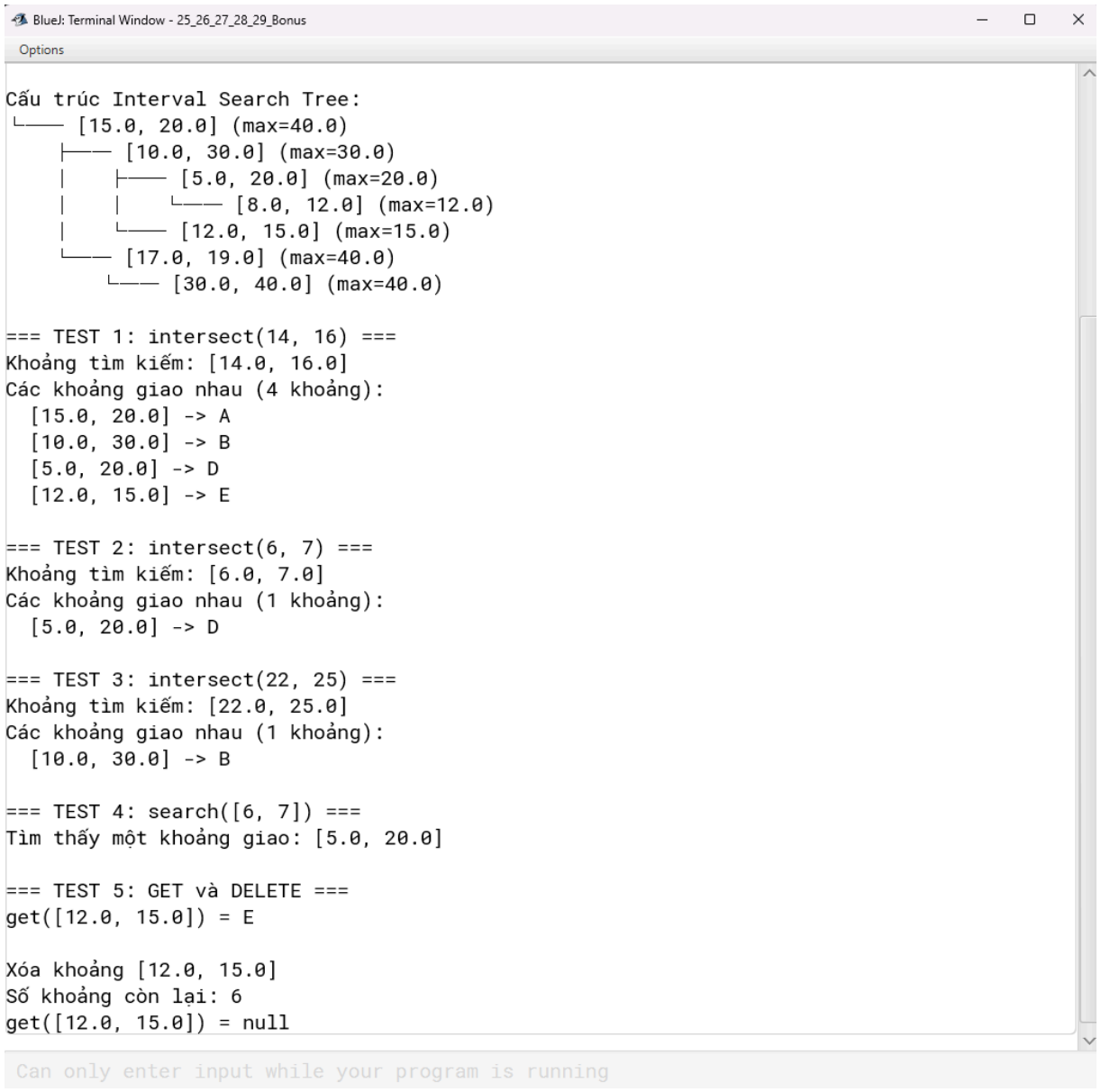
```
        Queue<Interval1D> queue = new Queue<>();
        intersect(root, interval, queue);
        return queue;
    }

    public Queue<Interval1D> intersect(Interval1D interval) {
        if (interval == null) throw new IllegalArgumentException("Interval
cannot be null");
        Queue<Interval1D> queue = new Queue<>();
        intersect(root, interval, queue);
        return queue;
    }

    private void intersect(Node x, Interval1D interval, Queue<Interval1D> queue)
{
        if (x == null) return;

        if (x.interval.intersects(interval)) {
            queue.enqueue(x.interval);
        }

        if (x.left != null && x.left.max >= interval.min()) {
            intersect(x.left, interval, queue);
        }

        if (x.right != null && interval.max() >= x.interval.min()) {
            intersect(x.right, interval, queue);
        }
    }

    public Interval1D search(Interval1D interval) {
        if (interval == null) throw new IllegalArgumentException("Interval
cannot be null");
        return search(root, interval);
    }

    private Interval1D search(Node x, Interval1D interval) {
        if (x == null) return null;

        if (x.interval.intersects(interval)) {
            return x.interval;
        }

        if (x.left != null && x.left.max >= interval.min()) {
            return search(x.left, interval);
        }

        return search(x.right, interval);
    }

    public Queue<Interval1D> intervals() {
        Queue<Interval1D> queue = new Queue<>();
        intervals(root, queue);
```

```
        return queue;
    }

    private void intervals(Node x, Queue<Interval1D> queue) {
        if (x == null) return;
        intervals(x.left, queue);
        queue.enqueue(x.interval);
        intervals(x.right, queue);
    }

    public void printTree() {
        System.out.println("Cấu trúc Interval Search Tree:");
        printTree(root, "", true);
    }

    private void printTree(Node x, String prefix, boolean isTail) {
        if (x == null) return;

        System.out.println(prefix + (isTail ? "└── " : "├── ") +
                           x.interval + " (max=" + x.max + ")");

        if (x.left != null || x.right != null) {
            if (x.left != null) {
                printTree(x.left, prefix + (isTail ? "    " : "│   "), x.right
== null);
            }
            if (x.right != null) {
                printTree(x.right, prefix + (isTail ? "    " : "│   "), true);
            }
        }
    }

    public static void main(String[] args) {
        System.out.println("=== BÀI TẬP 28: INTERVAL SEARCH TREE ===\n");

        IntervalSearchTree<String> ist = new IntervalSearchTree<>();

        // Thêm các khoảng
        System.out.println("Thêm các khoảng vào cây:");
        ist.put(new Interval1D(15, 20), "A");
        ist.put(new Interval1D(10, 30), "B");
        ist.put(new Interval1D(17, 19), "C");
        ist.put(new Interval1D(5, 20), "D");
        ist.put(new Interval1D(12, 15), "E");
        ist.put(new Interval1D(30, 40), "F");
        ist.put(new Interval1D(8, 12), "G");

        Queue<Interval1D> allIntervals = ist.intervals();
        int count = 1;
        for (Interval1D interval : allIntervals) {
            System.out.println("  " + count++ + ". " + interval +
                               " -> " + ist.get(interval));
        }
```

```
        System.out.println("\nTổng số khoảng: " + ist.size());
        System.out.println();
        ist.printTree();

        // TEST 1: Tìm tất cả khoảng giao với [14, 16]
        System.out.println("\n=== TEST 1: intersect(14, 16) ===");
        Queue<Interval1D> result1 = ist.intersect(14, 16);
        System.out.println("Khoảng tìm kiếm: [14.0, 16.0]");
        System.out.println("Các khoảng giao nhau (" + result1.size() + "
khoảng):");
        for (Interval1D interval : result1) {
            System.out.println("  " + interval + " -> " + ist.get(interval));
        }

        // TEST 2: Tìm tất cả khoảng giao với [6, 7]
        System.out.println("\n=== TEST 2: intersect(6, 7) ===");
        Queue<Interval1D> result2 = ist.intersect(6, 7);
        System.out.println("Khoảng tìm kiếm: [6.0, 7.0]");
        System.out.println("Các khoảng giao nhau (" + result2.size() + "
khoảng):");
        for (Interval1D interval : result2) {
            System.out.println("  " + interval + " -> " + ist.get(interval));
        }

        // TEST 3: Tìm tất cả khoảng giao với [22, 25]
        System.out.println("\n=== TEST 3: intersect(22, 25) ===");
        Queue<Interval1D> result3 = ist.intersect(22, 25);
        System.out.println("Khoảng tìm kiếm: [22.0, 25.0]");
        System.out.println("Các khoảng giao nhau (" + result3.size() + "
khoảng):");
        for (Interval1D interval : result3) {
            System.out.println("  " + interval + " -> " + ist.get(interval));
        }

        // TEST 4: Tìm một khoảng giao (search)
        System.out.println("\n=== TEST 4: search([6, 7]) ===");
        Interval1D query = new Interval1D(6, 7);
        Interval1D found = ist.search(query);
        if (found != null) {
            System.out.println("Tìm thấy một khoảng giao: " + found);
        } else {
            System.out.println("Không tìm thấy khoảng giao");
        }

        // TEST 5: Get và Delete
        System.out.println("\n=== TEST 5: GET và DELETE ===");
        Interval1D testInterval = new Interval1D(12, 15);
        System.out.println("get(" + testInterval + ") = " +
ist.get(testInterval));

        System.out.println("\nXóa khoảng " + testInterval);
        ist.delete(testInterval);
```

```
        System.out.println("Số khoảng còn lại: " + ist.size());
        System.out.println("get(" + testInterval + ") = " +
ist.get(testInterval));
    }
}
```

Kết quả:



Bài tập 29: Cài đặt lớp giao cắt hình chữ nhật cho n hình chữ nhật (điểm lo, điểm hi) tìm Queue<i, j> các cặp chữ nhật i giao cắt các cặp chữ nhật j

Code:

RectangleIntersection.java

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class RectangleIntersection {
```

```
    public static class Pair {
        public final int i;
        public final int j;

        public Pair(int i, int j) {
            if (i < j) {
                this.i = i;
                this.j = j;
            } else {
                this.i = j;
                this.j = i;
            }
        }

        @Override
        public boolean equals(Object obj) {
            if (!(obj instanceof Pair)) return false;
            Pair other = (Pair) obj;
            return this.i == other.i && this.j == other.j;
        }

        @Override
        public int hashCode() {
            return 31 * i + j;
        }

        @Override
        public String toString() {
            return "(" + i + ", " + j + ")";
        }
    }

    private static class Event implements Comparable<Event> {
        double x;           // Tọa độ x của sự kiện
        int type;           // 0: bắt đầu, 1: kết thúc
        int rectIndex;      // Chỉ số hình chữ nhật
        Interval1D yInterval; // Khoảng y của hình chữ nhật

        public Event(double x, int type, int rectIndex, Interval1D yInterval) {
            this.x = x;
            this.type = type;
            this.rectIndex = rectIndex;
            this.yInterval = yInterval;
        }

        @Override
        public int compareTo(Event other) {
            if (this.x != other.x) {
                return Double.compare(this.x, other.x);
            }
            return Integer.compare(this.type, other.type);
        }
    }
```

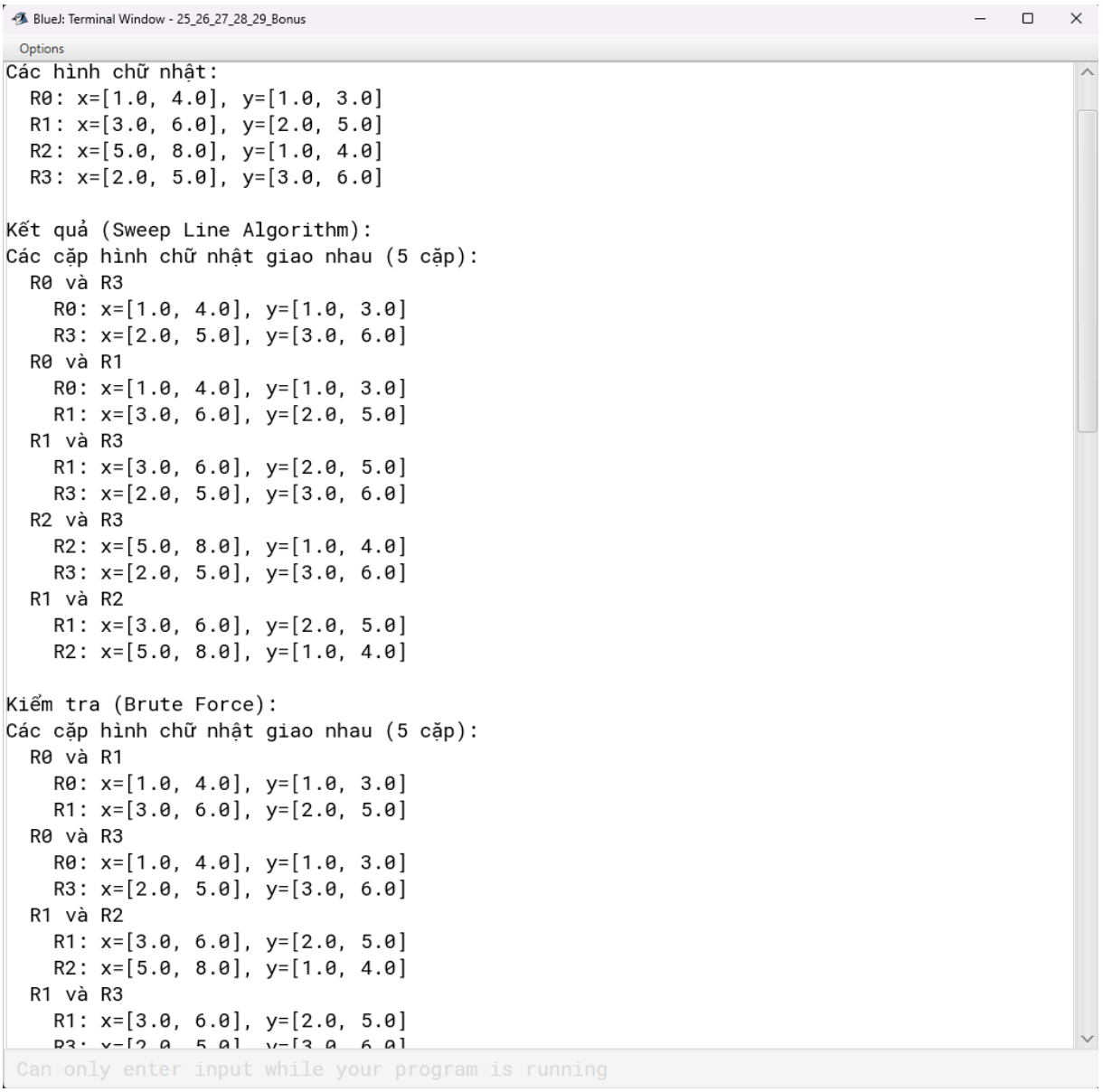
```
    public static Queue<Pair> findIntersections(Interval2D[] rectangles) {
        if (rectangles == null) throw new IllegalArgumentException("Rectangles
array cannot be null");

        int n = rectangles.length;
        Queue<Pair> result = new Queue<>();

        if (n <= 1) return result;

        List<Event> events = new ArrayList<>();

        for (int i = 0; i < n; i++) {
            Interval2D rect = rectangles[i];
            Interval1D xInterval = getXInterval(rect);
            Interval1D yInterval = getYInterval(rect);

            events.add(new Event(xInterval.min(), 0, i, yInterval));
            events.add(new Event(xInterval.max(), 1, i, yInterval));
        }

        Collections.sort(events);

        IntervalSearchTree<Integer> activeRects = new IntervalSearchTree<>();

        for (Event event : events) {
            if (event.type == 0) {
                Queue<Interval1D> intersecting =
activeRects.intersect(event.yInterval);

                for (Interval1D yInterval : intersecting) {
                    Integer otherIndex = activeRects.get(yInterval);
                    if (otherIndex != null) {
                        result.enqueue(new Pair(event.rectIndex, otherIndex));
                    }
                }

                activeRects.put(event.yInterval, event.rectIndex);

            } else {
                activeRects.delete(event.yInterval);
            }
        }

        return result;
    }

    public static Queue<Pair> findIntersectionsBruteForce(Interval2D[]
rectangles) {
        Queue<Pair> result = new Queue<>();
        int n = rectangles.length;

        for (int i = 0; i < n; i++) {
```

```
            for (int j = i + 1; j < n; j++) {
                if (rectangles[i].intersects(rectangles[j])) {
                    result.enqueue(new Pair(i, j));
                }
            }
        }

        return result;
    }

    private static Interval1D getXInterval(Interval2D rect) {
        String str = rect.toString();
        String xPart = str.split(" x ")[0];
        String[] bounds = xPart.replace("[", "").replace("]", "").split(", ");
        return new Interval1D(Double.parseDouble(bounds[0]),
Double.parseDouble(bounds[1]));
    }

    private static Interval1D getYInterval(Interval2D rect) {
        String str = rect.toString();
        String yPart = str.split(" x ")[1];
        String[] bounds = yPart.replace("[", "").replace("]", "").split(", ");
        return new Interval1D(Double.parseDouble(bounds[0]),
Double.parseDouble(bounds[1]));
    }

    private static void printRectangles(Interval2D[] rectangles) {
        for (int i = 0; i < rectangles.length; i++) {
            Interval1D xInterval = getXInterval(rectangles[i]);
            Interval1D yInterval = getYInterval(rectangles[i]);
            System.out.println("  R" + i + ": x=" + xInterval + ", y=" +
yInterval);
        }
    }

    private static void printPairs(Queue<Pair> pairs, Interval2D[] rectangles) {
        System.out.println("Các cặp hình chữ nhật giao nhau (" + pairs.size() +
" cặp):");

        if (pairs.isEmpty()) {
            System.out.println("  (không có)");
            return;
        }

        for (Pair p : pairs) {
            System.out.println("  R" + p.i + " và R" + p.j);
            Interval1D xi = getXInterval(rectangles[p.i]);
            Interval1D yi = getYInterval(rectangles[p.i]);
            Interval1D xj = getXInterval(rectangles[p.j]);
            Interval1D yj = getYInterval(rectangles[p.j]);
            System.out.println("    R" + p.i + ": x=" + xi + ", y=" + yi);
            System.out.println("    R" + p.j + ": x=" + xj + ", y=" + yj);
        }
```

```
    }

    public static void main(String[] args) {
        System.out.println("=== BÀI TẬP 29: TÌM CẶP HÌNH CHỮ NHẬT GIAO NHAU
===\n");

        System.out.println("=== TEST 1: Ví dụ cơ bản ===");
        Interval2D[] rects1 = {
            new Interval2D(new Interval1D(1, 4), new Interval1D(1, 3)),  // R0
            new Interval2D(new Interval1D(3, 6), new Interval1D(2, 5)),  // R1
            new Interval2D(new Interval1D(5, 8), new Interval1D(1, 4)),  // R2
            new Interval2D(new Interval1D(2, 5), new Interval1D(3, 6))   // R3
        };

        System.out.println("Các hình chữ nhật:");
        printRectangles(rects1);

        System.out.println("\nKết quả (Sweep Line Algorithm):");
        Queue<Pair> result1 = findIntersections(rects1);
        printPairs(result1, rects1);

        System.out.println("\nKiểm tra (Brute Force):");
        Queue<Pair> check1 = findIntersectionsBruteForce(rects1);
        printPairs(check1, rects1);

        System.out.println("\n=== TEST 2: Trường hợp phức tạp hơn ===");
        Interval2D[] rects2 = {
            new Interval2D(new Interval1D(0, 3), new Interval1D(0, 3)),   // R0
            new Interval2D(new Interval1D(2, 5), new Interval1D(1, 4)),   // R1
            new Interval2D(new Interval1D(4, 7), new Interval1D(2, 5)),   // R2
            new Interval2D(new Interval1D(1, 4), new Interval1D(3, 6)),   // R3
            new Interval2D(new Interval1D(6, 9), new Interval1D(1, 4)),   // R4
            new Interval2D(new Interval1D(3, 6), new Interval1D(4, 7))    // R5
        };

        System.out.println("Số lượng hình chữ nhật: " + rects2.length);

        Queue<Pair> result2 = findIntersections(rects2);
        printPairs(result2, rects2);

        System.out.println("\n=== TEST 3: Không có giao nhau ===");
        Interval2D[] rects3 = {
            new Interval2D(new Interval1D(0, 1), new Interval1D(0, 1)),
            new Interval2D(new Interval1D(2, 3), new Interval1D(2, 3)),
            new Interval2D(new Interval1D(4, 5), new Interval1D(4, 5))
        };

        System.out.println("Các hình chữ nhật:");
        printRectangles(rects3);

        Queue<Pair> result3 = findIntersections(rects3);
        printPairs(result3, rects3);
```

```
        System.out.println("\n=== TEST 4: Tất cả giao nhau ===");
        Interval2D[] rects4 = {
            new Interval2D(new Interval1D(1, 5), new Interval1D(1, 5)),
            new Interval2D(new Interval1D(2, 6), new Interval1D(2, 6)),
            new Interval2D(new Interval1D(3, 7), new Interval1D(3, 7))
        };

        System.out.println("Các hình chữ nhật:");
        printRectangles(rects4);

        Queue<Pair> result4 = findIntersections(rects4);
        printPairs(result4, rects4);

        System.out.println("\n=== TEST 5: So sánh hiệu suất ===");
        int n = 100;
        Interval2D[] largeTest = new Interval2D[n];
        for (int i = 0; i < n; i++) {
            double x1 = Math.random() * 100;
            double x2 = x1 + Math.random() * 20;
            double y1 = Math.random() * 100;
            double y2 = y1 + Math.random() * 20;
            largeTest[i] = new Interval2D(new Interval1D(x1, x2), new
Interval1D(y1, y2));
        }

        System.out.println("Kiểm tra với " + n + " hình chữ nhật...");

        long start1 = System.currentTimeMillis();
        Queue<Pair> sweepResult = findIntersections(largeTest);
        long time1 = System.currentTimeMillis() - start1;

        long start2 = System.currentTimeMillis();
        Queue<Pair> bruteResult = findIntersectionsBruteForce(largeTest);
        long time2 = System.currentTimeMillis() - start2;

        System.out.println("Sweep Line: " + sweepResult.size() + " cặp, thời
gian: " + time1 + "ms");
        System.out.println("Brute Force: " + bruteResult.size() + " cặp, thời
gian: " + time2 + "ms");
        System.out.println("Kết quả khớp: " + (sweepResult.size() ==
bruteResult.size()));
    }
}
```

Kết quả:


BlueJ: Terminal Window - 25_26_27_28_29_Bonus
Options
Các hình chữ nhật:
R0: x=[1.0, 4.0], y=[1.0, 3.0]
R1: x=[3.0, 6.0], y=[2.0, 5.0]
R2: x=[5.0, 8.0], y=[1.0, 4.0]
R3: x=[2.0, 5.0], y=[3.0, 6.0]
Kết quả (Sweep Line Algorithm):
Các cặp hình chữ nhật giao nhau (5 cặp):
R0 và R3
R0: x=[1.0, 4.0], y=[1.0, 3.0]
R3: x=[2.0, 5.0], y=[3.0, 6.0]
R0 và R1
R0: x=[1.0, 4.0], y=[1.0, 3.0]
R1: x=[3.0, 6.0], y=[2.0, 5.0]
R1 và R3
R1: x=[3.0, 6.0], y=[2.0, 5.0]
R3: x=[2.0, 5.0], y=[3.0, 6.0]
R2 và R3
R2: x=[5.0, 8.0], y=[1.0, 4.0]
R3: x=[2.0, 5.0], y=[3.0, 6.0]
R1 và R2
R1: x=[3.0, 6.0], y=[2.0, 5.0]
R2: x=[5.0, 8.0], y=[1.0, 4.0]
Kiểm tra (Brute Force):
Các cặp hình chữ nhật giao nhau (5 cặp):
R0 và R1
R0: x=[1.0, 4.0], y=[1.0, 3.0]
R1: x=[3.0, 6.0], y=[2.0, 5.0]
R0 và R3
R0: x=[1.0, 4.0], y=[1.0, 3.0]
R3: x=[2.0, 5.0], y=[3.0, 6.0]
R1 và R2
R1: x=[3.0, 6.0], y=[2.0, 5.0]
R2: x=[5.0, 8.0], y=[1.0, 4.0]
R1 và R3
R1: x=[3.0, 6.0], y=[2.0, 5.0]
Can only enter input while your program is running