TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA Công Nghệ Thông Tin
BỘ MÔN: Công Nghệ Phần Mềm

ĐỀ THI VÀ BÀI LÀM

Tên học phần: Math for CS
Mã học phần: Hình thức thi: *Tự luận có giám sát*
Đề số: **01/02** Thời gian làm bài: 90 phút *(không kể thời gian chép/phát đề)*
Được sử dụng tài liệu khi làm bài.

Họ tên: Nguyễn Thành Trung **Lớp**: 22T_KHDL **MSSV**: 102220044

Sinh viên làm bài trực tiếp trên tệp này, lưu tệp với định dạng MSSV_HọTên.pdf và nộp bài thông qua MSTeam:

Câu 1 (4 điểm): Viết chương trình nhập mã số sinh viên của bản thân(N) , thực hiện các công việc sau:

a) Hãy viết hàm để phân tích N ra hữu hạn các số nguyên tố

 # Trả lời: Dán code vào bên dưới:

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

public class PrimeFactorization {
    public static List<Integer> primeFactorization(int N) {
        List<Integer> factors = new ArrayList<>();
        int divisor = 2;
        while (divisor * divisor <= N) {
            while (N % divisor == 0) {
                factors.add(divisor);
                N /= divisor;
            }
            divisor++;
        }
        if (N > 1) {
            factors.add(N);
        }

        return factors;
    }

    public static void main(String[] args) {
        int number = 102220044;
        List<Integer> primeFactors = primeFactorization(number);
        System.out.println("Các thừa số nguyên tố của " + number + " là: " + primeFactors);
    }
}
```

Trả lời: Dán kết quả thực thi vào bên dưới:

```
Caused by: java.lang.ClassNotFoundException: base.src.hello
PS D:\java> cd base
PS D:\java\base> cd src
PS D:\java\base\src> java .\hello.java
Các th?a s? nguyên t? c?a 102220044 là: [2, 2, 3, 139, 61283]
PS D:\java\base\src>
```

b) Áp dụng câu a) hãy viết hàm tính tổng các ước của N

Trả lời: Dán code vào bên dưới

```
package base.src;
import java.util.ArrayList;
import java.util.List;

public class hello {
    public static int sumOfDivisors(int N) {
        int sum = 0;
        for (int i = 1; i <= Math.sqrt(N); i++) {
            if (N % i == 0) {
                sum += i;

                if (i != N / i) {
                    sum += N / i;
                }
            }
        }

        return sum;
    }

    public static void main(String[] args) {
        int number = 102220044;
        System.out.println("Tổng các ước của " + number + " là: " + sumOfDivisors(number));
    }
}
```

```
PS D:\java\base\src> java .\hello.java
T?ng các ??c c?a 102220044 là: 240233280
PS D:\java\base\src>
```

c) Áp dụng câu a) hãy viết hàm tính tích các ước của N

Trả lời: Dán code vào bên dưới

```
package base.src;
import java.math.BigInteger;

public class Hello {
    public static BigInteger productOfDivisors(long N) {
        BigInteger product = BigInteger.ONE;
        for (long i = 1; i <= Math.sqrt(N); i++) {
            if (N % i == 0) {
                product = product.multiply(BigInteger.valueOf(i));
                if (i != N / i) {
```

```
                product = product.multiply(BigInteger.valueOf(N / i));
            }
        }
    }

    return product;
}

public static void main(String[] args) {
    long number = 102220044;
    System.out.println("Tích các ước của " + number + " là: " + productOfDivisors(number));
}
}
```

```
Tích các ??c c?a 102220044 là: 13014658075720725312195837318281224608748549340017924732872211447631775453361429841038213157748736
PS D:\java\base\src>
```

d) Hãy viết hàm để xác định các số hoàn hảo không vượt quá N

 # Trả lời: Dán code vào bên dưới

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

public class PerfectNumber {
    // Hàm tính tổng các ước thực sự của số nguyên dương N (không bao gồm N)
    public static int sumOfProperDivisors(int N) {
        int sum = 0;

        // Duyệt qua từ 1 đến sqrt(N) để tìm các ước của N
        for (int i = 1; i <= Math.sqrt(N); i++) {
            if (N % i == 0) { // Nếu i là ước của N
                sum += i; // Thêm i vào tổng

                // Thêm cả N / i vào tổng nếu i không phải là 1 và N / i khác i
                if (i != 1 && i != N / i) {
                    sum += N / i;
                }
            }
        }

        return sum;
    }

    // Hàm xác định các số hoàn hảo không vượt quá N
    public static List<Integer> perfectNumbersUpTo(int N) {
        List<Integer> perfectNumbers = new ArrayList<>();

        // Duyệt qua từ 2 đến N để tìm các số hoàn hảo
        for (int i = 2; i <= N; i++) {
            if (sumOfProperDivisors(i) == i) { // Nếu tổng các ước thực sự của i bằng chính nó
                perfectNumbers.add(i); // Thêm i vào danh sách số hoàn hảo
            }
        }

        return perfectNumbers;
    }

    public static void main(String[] args) {
        int N = 102220044; // Giới hạn trên cho các số hoàn hảo
        List<Integer> perfectNumbers = perfectNumbersUpTo(N);

        // In ra các số hoàn hảo
        System.out.println("Các số hoàn hảo không vượt quá " + N + " là:");
```

```
        for (int number : perfectNumbers) {
            System.out.println(number);
        }
    }
}
```

 # Trả lời: Dán kết quả thực thi vào bên dưới:

```
Các số hoàn hảo không vượt quá 102220044 là:
6
28
496
8128
```

Câu 2 (3 *điểm*): Phân rã ma trận A (nếu mã số sinh viên là số lẽ thì thực hiện phân rã Cholesky, còn mã sinh viên là chẳn thì thực hiện phân rã SVD)

a) Hãy trình bày điều kiện của ma trân A để thực hiện phân rã Cholesky / SVD.

 # Trả lời: Trình bày điều kiện vào bên dưới:

Ma trận phải là ma trận hình chữ nhật:

Ma trận có thể là ma trận vuông (m x m) hoặc ma trận hình chữ nhật (m x n), với m là số hàng và n là số cột. Tuy nhiên, nếu ma trận là hình vuông, nó sẽ có các đặc điểm đặc biệt.

Ma trận thực:

SVD có thể được thực hiện trên các ma trận thực (real matrices) hoặc các ma trận phức (complex matrices). Tuy nhiên, trong trường hợp này, ta thường xem xét các ma trận thực.

Phân tích có thể không xác định:

Nếu ma trận không có đầy đủ hạng (rank), tức là một số dòng hoặc cột là tổ hợp tuyến tính của các dòng hoặc cột còn lại, SVD vẫn có thể được thực hiện, nhưng số giá trị riêng không bằng số hàng hoặc số cột.

Ma trận có hạng không âm:

Hạng của ma trận (rank) phải không âm và được xác định thông qua các giá trị riêng (singular values) trong SVD.

b) Hãy dùng sơ đồ khối để miêu tả thuật toán phân rã Cholesky / SVD.

 # Trả lời: Dán sơ đồ khối vào đây vào bên dưới (khuyến khích dùng io draw)

c) Viết hàm để thực thi phân rã Cholesky / SVD.

 # Trả lời: Dán code vào bên dưới

```
package hehe;

import java.util.Scanner;
import org.apache.commons.math3.linear.*;

public class Main {
    public static void svd(double[][] a) {
        RealMatrix matrix = new Array2DRowRealMatrix(a);

        // Phân ra ký di (SVD)
        SingularValueDecomposition svd = new SingularValueDecomposition(matrix);
```

```
    // Lấy ma trận U, Sigma và V
    RealMatrix U = svd.getU();
    RealMatrix Sigma = svd.getS();
    RealMatrix V = svd.getV();

    // In U
    System.out.println(" U:");
    printMatrix(U);

    // In Sigma
    System.out.println("\n Sigma:");
    printMatrix(Sigma);

    // In V
    System.out.println("\n V:");
    printMatrix(V);

    // Kiểm tra bằng cách nhân lại
    RealMatrix V_T = V.transpose();
    RealMatrix A_reconstructed = U.multiply(Sigma).multiply(V_T);

    System.out.println("\nMa trận A (từ U * Sigma * V^T):");
    printMatrix(A_reconstructed);
}

private static void printMatrix(RealMatrix matrix) {
    int rowCount = matrix.getRowDimension();
    int colCount = matrix.getColumnDimension();
    for (int i = 0; i < rowCount; i++) {
        for (int j = 0; j < colCount; j++) {
            System.out.printf("%.2f ", matrix.getEntry(i, j));
            System.out.print(" ");
        }
        System.out.println();
    }
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Nhập vào kích thước m và n của ma trận (m*n):");
    int m = scanner.nextInt();
    int n = scanner.nextInt();
    double a[][] = new double[m][n];

    System.out.println("Nhập các phần tử của ma trận:");
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            a[i][j] = scanner.nextDouble();
        }
    }

    // Gọi hàm phân tích SVD
    svd(a);
```

```
    }
}
```

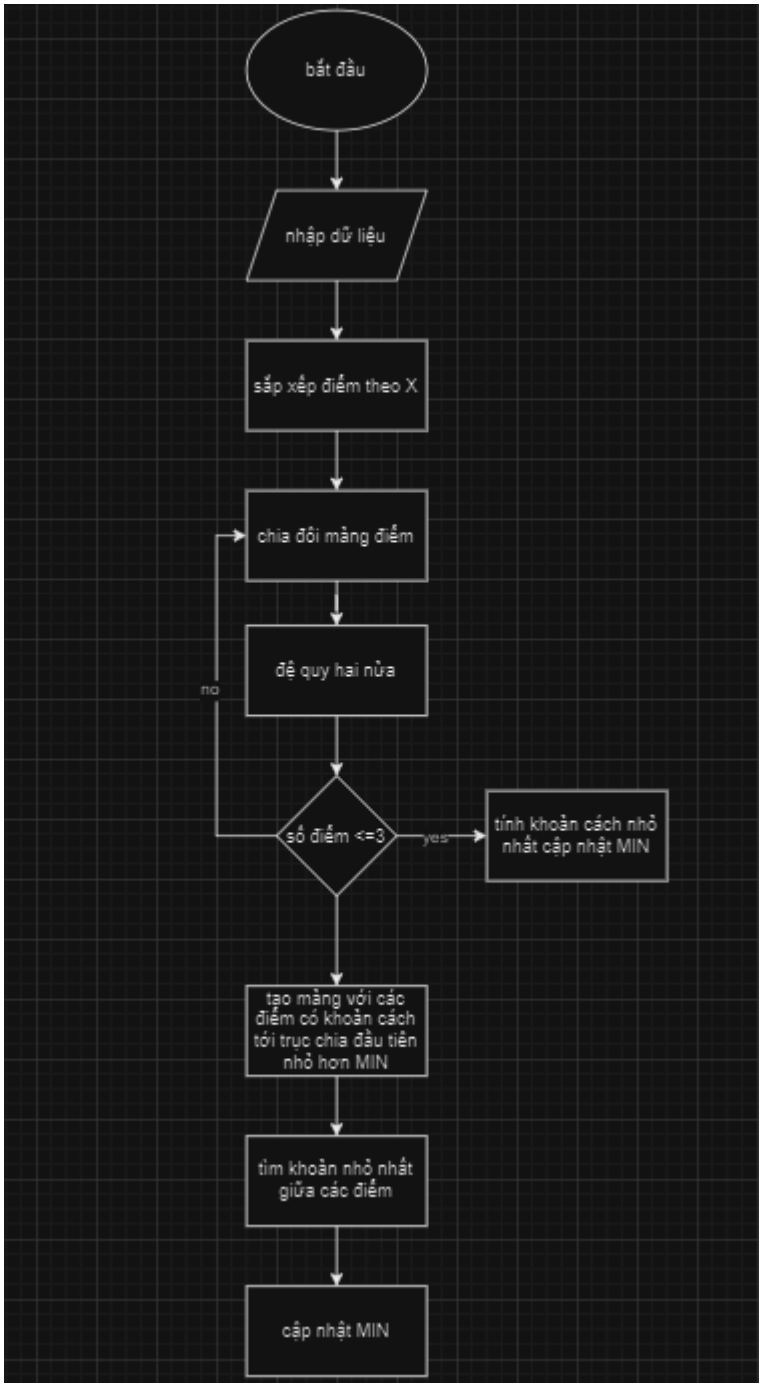
Trả lời: Minh hoạ kết quả vào bên dưới

```
Nh?p vào kích th??c m và n c?a ma tr?n (m*n):
3
2
Nh?p các ph?n t? c?a ma tr?n:
1
2
3
4
5
6
 U:
0.23  -0.88
0.52  -0.24
0.82  0.40

 Sigma:
9.53  0.00
0.00  0.51

 V:
0.62  0.78
0.78  -0.62

Ma tr?n A (t? U * Sigma * V^T):
1.00  2.00
3.00  4.00
5.00  6.00
```

Câu 2 (3 *điểm*): Cho không gian Oxy và 7 điểm tương ứng như hình vẽ dưới:

a) *Dùng sơ đồ khối để mô tả thuật toán xác định khoảng cách ngắn nhất giữa 2 điểm trong N điểm(thuật toán đảm bảo nlog(n)*

 # Trả lời: Dán sơ đồ khối vào bên dưới:



b) *Viết chương trình dạng hàm mô tả thuật toán ở câu a)*

 # Trả lời: viết câu trả lời vào bên dưới:

```
import java.util.Arrays;

import java.util.Comparator;



public class ClosestPair {
```

```
    static class Point {
        double x, y;

        public Point(double x, double y) {
            this.x = x;
            this.y = y;
        }
    }

    // Hàm tính khoảng cách giữa hai điểm
    static double distance(Point p1, Point p2) {
        return Math.sqrt((p1.x - p2.x) * (p1.x - p2.x) + (p1.y - p2.y) * (p1.y - p2.y));
    }

    // Hàm tìm khoảng cách nhỏ nhất trong một dải giữa
    static double stripClosest(Point[] strip, int size, double d) {
        double min = d; // khoảng cách ban đầu

        Arrays.sort(strip, 0, size, Comparator.comparingDouble(p -> p.y)); // sắp xếp theo tung độ (y)

        for (int i = 0; i < size; i++) {
            for (int j = i + 1; j < size && (strip[j].y - strip[i].y) < min; j++) {
                if (distance(strip[i], strip[j]) < min) {
                    min = distance(strip[i], strip[j]);
                }
            }
        }
```

```
    return min;
}

// Hàm đệ quy để tìm khoảng cách nhỏ nhất
static double closestUtil(Point[] points, int left, int right) {
    if (right - left <= 3) { // nếu ít hơn hoặc bằng 3 điểm, sử dụng brute-force
        double minDist = Double.MAX_VALUE;
        for (int i = left; i < right; i++) {
            for (int j = i + 1; j <= right; j++) {
                if (distance(points[i], points[j]) < minDist) {
                    minDist = distance(points[i], points[j]);
                }
            }
        }
        return minDist;
    }

    int mid = left + (right - left) / 2;
    Point midPoint = points[mid];

    // Tìm khoảng cách nhỏ nhất bên trái và bên phải
    double dl = closestUtil(points, left, mid);
    double dr = closestUtil(points, mid + 1, right);

    double d = Math.min(dl, dr); // khoảng cách nhỏ nhất giữa hai nửa

    // Tạo một dải chứa các điểm gần đường phân chia
```

```
    Point[] strip = new Point[right - left + 1];
    int j = 0;
    for (int i = left; i <= right; i++) {
        if (Math.abs(points[i].x - midPoint.x) < d) {
            strip[j] = points[i];
            j++;
        }
    }

    // Kiểm tra khoảng cách nhỏ nhất trong dải
    return Math.min(d, stripClosest(strip, j, d));
}

// Hàm chính để tìm khoảng cách nhỏ nhất
static double closest(Point[] points, int n) {
    Arrays.sort(points, Comparator.comparingDouble(p -> p.x)); // sắp xếp theo hoành độ (x)
    return closestUtil(points, 0, n - 1);
}

public static void main(String[] args) {
    Point[] points = {
        new Point(2,1), new Point(4,3),
        new Point(1,6), new Point(4,5),
        new Point(3,4), new Point(5,2),new Point(5, 7)
    };

    int n = points.length;
    System.out.println("Khoảng cách nhỏ nhất là " + closest(points, n));
```

```
    }

}
```

Trả lời: Dán kết quả minh hoạ 7 điểm đã cho

```
PS D:\java> cd base\src
PS D:\java\base\src> java .\ClosestPair.java
Kho?ng cách nh? nh?t là 1.4142135623730951
PS D:\java\base\src>
```

Đà Nẵng, ngày 04 tháng 10 năm 2024

GIẢNG VIÊN BIÊN SOẠN ĐỀ THI

TRƯỞNG BỘ MÔN
(đã duyệt)