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: ……Ngô Quốc Anh…… **Lớp**:……22T_KHDL…… **MSSV**:……102220002…….

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:

```
void factor(int n)
{
   for (int i = 2; i <= sqrt(n); i++)
   {
      int mu = 0;
      while (n % i == 0)
      {
         n /= i;
         mu++;
      }
      if (mu != 0)
      {
         cout << i << "^" << mu;
         if (n > 1)
         {
            cout << " x ";
         }
      }
   }
   if (n > 1)
   {
      cout << n;
   }
}
```

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

```
PS C:\Users\admin\Documents\Toan_ung_dung_cntt> .\a.exe
Nhap N = 102220002
2^1 x 3^3 x 73^1 x 25931
```

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

```
# Trả lời: Dán code vào bên dưới
int tongCacUoc(int n) {
    int sum = 0;
    for (int i = 1; i * i <= n; i++) {
        if (n % i == 0) {
            sum += i;
            if (i != n / i) {
                sum += n / i;
            }
        }
    }
    return sum;
}
```

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

```
# Trả lời: Dán code vào bên dưới
string multiply(string num1, int num2) {
    if (num2 == 0 || num1 == "0") return "0";
    int carry = 0;
    string result = "";

    for (int i = num1.length() - 1; i >= 0; --i) {
        int product = (num1[i] - '0') * num2 + carry;
        result = char(product % 10 + '0') + result;
        carry = product / 10;
    }

    while (carry) {
        result = char(carry % 10 + '0') + result;
        carry /= 10;
    }

    return result;
}

string tichcacuoctu1denN(int N) {
    vector<string> dp(N + 1, "1");

    for (int i = 1; i <= N; ++i) {
        for (int j = i; j <= N; j += i) {
            dp[j] = multiply(dp[j], i);
        }
    }

    return dp[N];
```

```
}
```

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

 # Trả lời: Dán code vào bên dưới

```
bool isPrime(int num) {
    if (num <= 1) return false;
    for (int i = 2; i * i <= num; ++i) {
        if (num % i == 0) return false;
    }
    return true;
}

vector<long long> findPerfectNumbers(long long N) {
    vector<long long> perfectNumbers;

    for (int p = 2; p < 32; ++p) {
        if (isPrime(p)) {
            long long mersenne = (1LL << p) - 1;
            long long perfect = (1LL << (p - 1)) * mersenne;

            if (perfect <= N) {
                perfectNumbers.push_back(perfect);
            }
        }
    }

    return perfectNumbers;
}
```

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

```
PS C:\Users\admin\Documents\Toan_ung_dung_cntt> .\a.exe
Cac so hoan hao khong vuot qua 102220002 la: 6 28 496 8128 2096128 33550336
```

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)

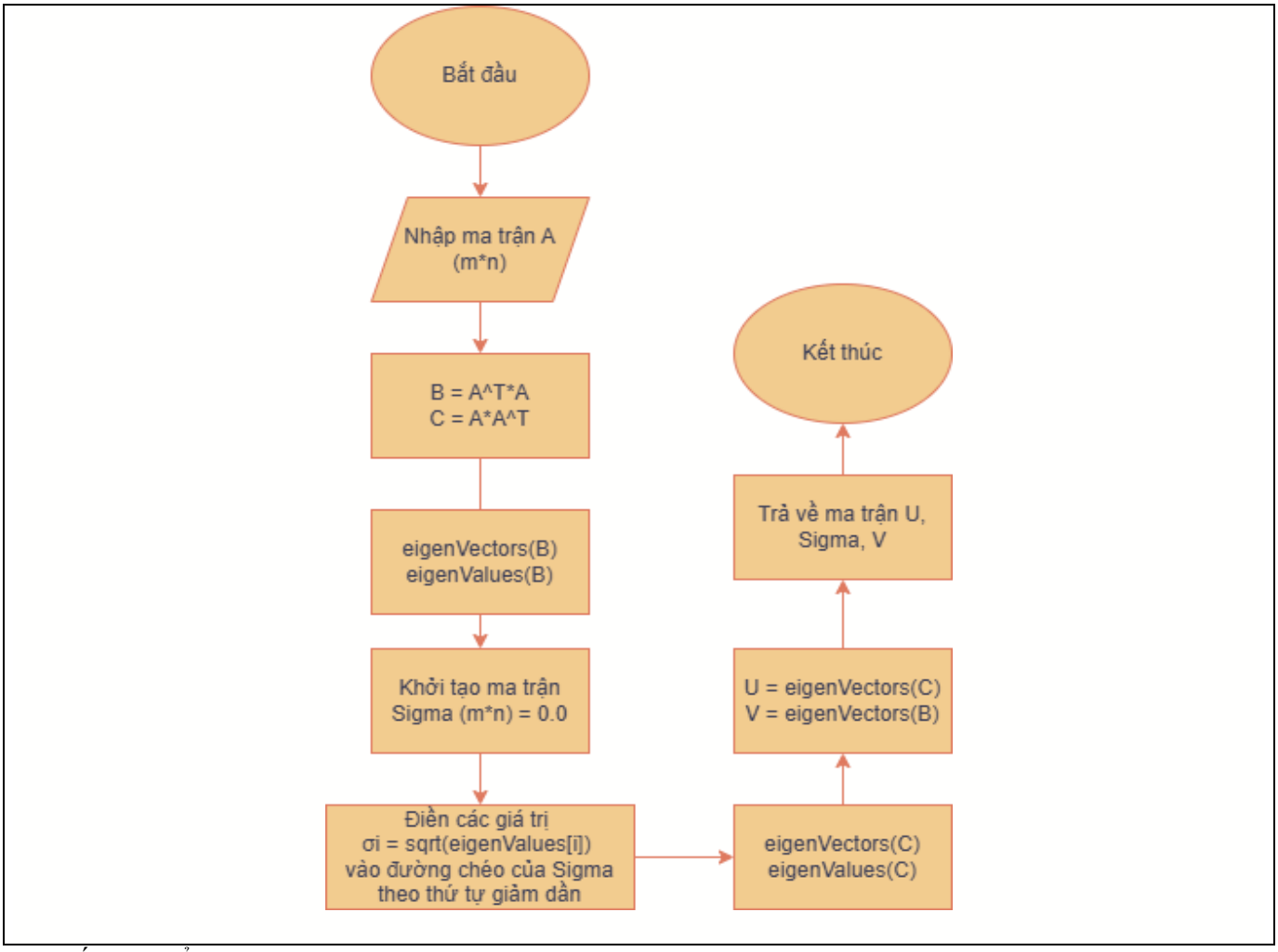
a) Hãy trình bày điều kiện của ma trân A để thực hiện phân rã Cholesky / SVD.

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

- Điều kiện phân rã SVD: Phân rã SVD có thể thực hiện trên bất kỳ ma trận nào, không cần phải đối xứng hay xác định dương. Tuy nhiên, nếu ma trận là ma trận số phức, việc phân rã SVD cần phải sử dụng 1 thuật toán đặc biệt.

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
#include <iostream>
#include <vector>
#include <cmath>
#include <algorithm>

using namespace std;

typedef vector<vector<double>> Matrix;

Matrix transpose(const Matrix& A) {
    int rows = A.size();
    int cols = A[0].size();
    Matrix AT(cols, vector<double>(rows));
    for (int i = 0; i < rows; ++i)
        for (int j = 0; j < cols; ++j)
            AT[j][i] = A[i][j];
    return AT;
}

Matrix multiply(const Matrix& A, const Matrix& B) {
    int rows = A.size();
    int cols = B[0].size();
    int inner = B.size();
```

```
    Matrix C(rows, vector<double>(cols, 0.0));
    for (int i = 0; i < rows; ++i)
        for (int j = 0; j < cols; ++j)
            for (int k = 0; k < inner; ++k)
                C[i][j] += A[i][k] * B[k][j];
    return C;
}

void jacobiMethod(const Matrix& A, Matrix& eigenVectors, vector<double>& eigenValues) {
    int n = A.size();
    eigenVectors = Matrix(n, vector<double>(n, 0.0));
    eigenValues = vector<double>(n, 0.0);

    for (int i = 0; i < n; ++i)
        eigenVectors[i][i] = 1.0;

    Matrix B = A;
    const int maxIterations = 100;
    const double tolerance = 1e-10;

    for (int iter = 0; iter < maxIterations; ++iter) {
        int p = 0, q = 1;
        double maxOffDiag = fabs(B[p][q]);
        for (int i = 0; i < n; ++i) {
            for (int j = i + 1; j < n; ++j) {
                if (fabs(B[i][j]) > maxOffDiag) {
                    maxOffDiag = fabs(B[i][j]);
                    p = i;
                    q = j;
                }
            }
        }

        if (maxOffDiag < tolerance)
            break;

        double theta = (B[q][q] - B[p][p]) / (2.0 * B[p][q]);
        double t = (theta >= 0 ? 1.0 : -1.0) / (fabs(theta) + sqrt(theta * theta + 1.0));
        double c = 1.0 / sqrt(t * t + 1.0);
        double s = t * c;

        for (int i = 0; i < n; ++i) {
            double temp = c * B[i][p] - s * B[i][q];
            B[i][q] = s * B[i][p] + c * B[i][q];
            B[i][p] = temp;
        }

        for (int i = 0; i < n; ++i) {
            double temp = c * B[p][i] - s * B[q][i];
            B[q][i] = s * B[p][i] + c * B[q][i];
            B[p][i] = temp;
        }
```

```
        for (int i = 0; i < n; ++i) {
            double temp = c * eigenVectors[i][p] - s * eigenVectors[i][q];
            eigenVectors[i][q] = s * eigenVectors[i][p] + c * eigenVectors[i][q];
            eigenVectors[i][p] = temp;
        }
    }

    for (int i = 0; i < n; ++i)
        eigenValues[i] = B[i][i];
}

void svd(const Matrix& A, Matrix& U, vector<double>& S, Matrix& V) {
    Matrix AT = transpose(A);
    Matrix ATA = multiply(AT, A);

    jacobiMethod(ATA, V, S);

    vector<pair<double, vector<double>>> eigenPairs;
    for (int i = 0; i < S.size(); ++i)
        eigenPairs.push_back({S[i], V[i]});

    sort(eigenPairs.rbegin(), eigenPairs.rend(), [](const auto& a, const auto& b) {
        return a.first < b.first;
    });

    for (int i = 0; i < S.size(); ++i) {
        S[i] = sqrt(eigenPairs[i].first);
        V[i] = eigenPairs[i].second;
    }

    Matrix AV = multiply(A, V);
    U = Matrix(A.size(), vector<double>(V.size()));
    for (int i = 0; i < U.size(); ++i)
        for (int j = 0; j < U[0].size(); ++j)
            U[i][j] = AV[i][j] / (S[j] + 1e-10);
}

Matrix createSigmaMatrix(const vector<double>& S, int m, int n) {
    Matrix Sigma(m, vector<double>(n, 0.0));
    for (int i = 0; i < min(m, n); ++i) {
        Sigma[i][i] = S[i];
    }
    return Sigma;
}

int main() {
    int n, m; cin >> m >> n;
    Matrix A(n, vector<double>(n));
    for (int i = 0; i < m; ++i)
        for (int j = 0; j < n; ++j)
            cin >> A[i][j];
```

```
    Matrix U, V;
    vector<double> S;

    svd(A, U, S, V);

    cout << "Matrix U:" << endl;
    for (const auto& row : U) {
        for (double val : row)
            cout << val << " ";
        cout << endl;
    }

    cout << "Singular values:" << endl;
    for (double val : S)
        cout << val << " ";
    cout << endl;

    cout << "Matrix V:" << endl;
    for (const auto& row : V) {
        for (double val : row)
            cout << val << " ";
        cout << endl;
    }
    Matrix Sigma = createSigmaMatrix(S, m, n);
    cout << "Matrix Sigma:" << endl;
    for (const auto& row : Sigma) {
        for (double val : row)
            cout << val << " ";
        cout << endl;
    }
    return 0;
}
```

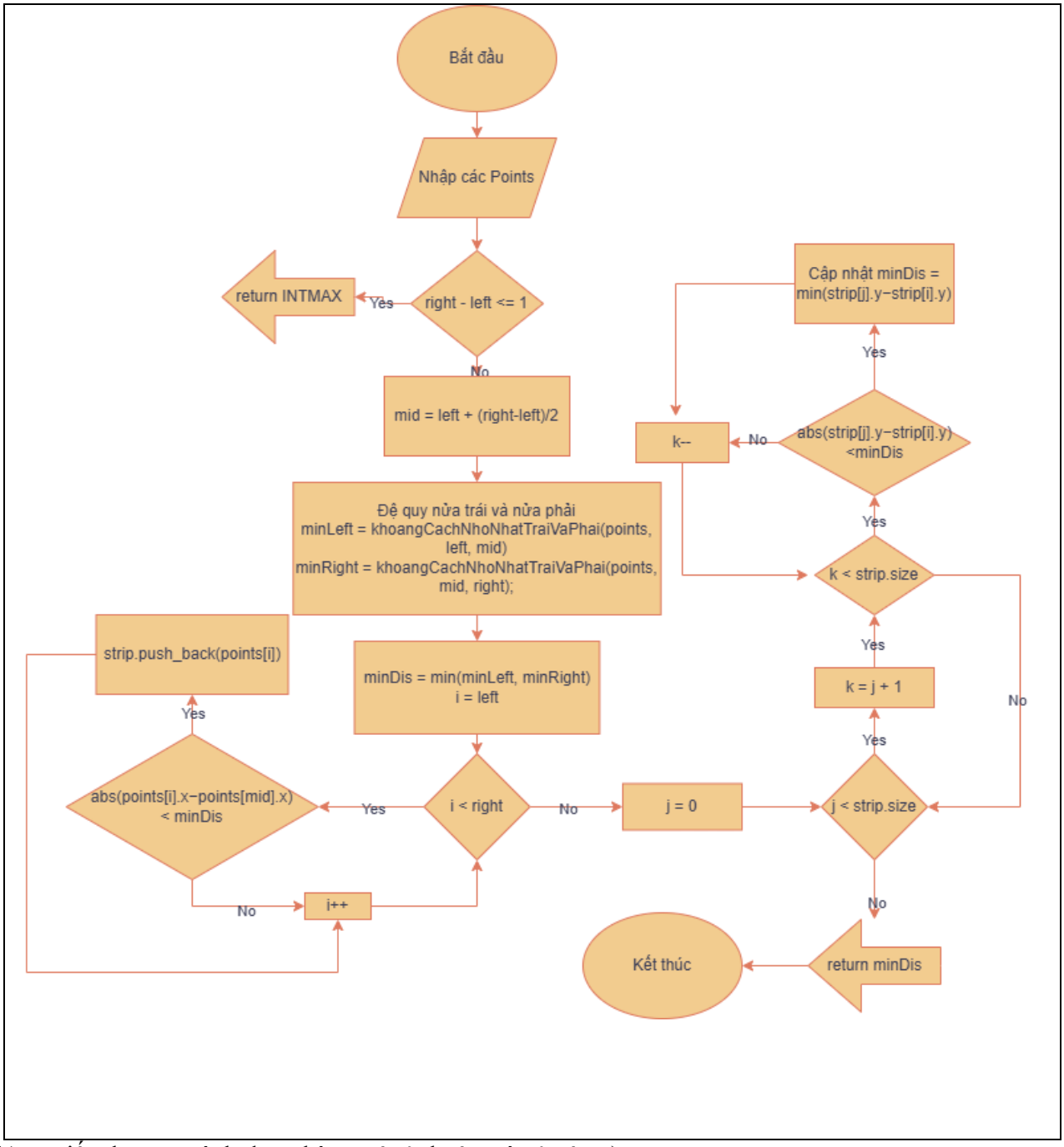
Trả lời: Minh hoạ kết quả vào bên dưới

```
PS C:\Users\admin\Documents\Toan_ung_dung_cntt> .\svd.exe
3 2
2 3
1 -2
5 4
Matrix U:
0.524046 1.63846
-0.5826 0.0866935
Singular values:
3.82843 1.82843
Matrix V:
-0.382683 0.92388
0.92388 0.382683
Matrix Sigma:
3.82843 0
0 1.82843
0 0
```

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:

```
#include <iostream>

#include <vector>

#include <algorithm>

#include <cmath>


using namespace std;
```

```
struct Point {
    int x;
    int y;

    Point(int x, int y) : x(x), y(y) {}

    bool operator<(const Point& other) const {
        return x < other.x || (x == other.x && y < other.y);
    }
};

double khoangCachEuclidean(const Point& a, const Point& b) {
    return sqrt(pow(a.x - b.x, 2) + pow(a.y - b.y, 2));
}

double khoangCachNhoNhatTraiVaPhai(vector<Point>& points, int left, int right) {
    if (right - left <= 1) {
        return INT_MAX;
    }

    int mid = left + (right - left) / 2;
    double minLeft = khoangCachNhoNhatTraiVaPhai(points, left, mid);
    double minRight = khoangCachNhoNhatTraiVaPhai(points, mid, right);

    double minDistance = min(minLeft, minRight);

    vector<Point> strip;
    for (int i = left; i < right; i++) {
```

```
        if (abs(points[i].x - points[mid].x) < minDistance) {
            strip.push_back(points[i]);
        }
    }

    sort(strip.begin(), strip.end(), [](const Point& a, const Point& b) {
        return a.y < b.y;
    });

    for (int i = 0; i < strip.size(); i++) {
        for (int j = i + 1; j < strip.size(); j++) {
            if (abs(strip[j].y - strip[i].y) >= minDistance) {
                break;
            }
            minDistance = min(minDistance, khoangCachEuclidean(strip[i], strip[j]));
        }
    }

    return minDistance;
}

double khoangCachNhoNhat(vector<Point>& points) {
    sort(points.begin(), points.end());
    return khoangCachNhoNhatTraiVaPhai(points, 0, points.size());
}

int main() {
    vector<Point> points = {
```

```
        Point(1, 2),
        Point(2, 5),
        Point(3, 4),
        Point(4, 3),
        Point(5, 4),
        Point(6, 1),
        Point(7, 5),
    };
    cout << "Khoang cach nho nhat: " << khoangCachNhoNhat(points) << endl;
    return 0;
}
```

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

```
PS C:\Users\admin\Documents\Toan_ung_dung_cntt> .\a.exe
Khoang cach nho nhat: 1.41421
```

Đà Nẵng, ngày 04 tháng 10 năm 2024

GIẢNG VIÊN BIÊN SOẠN ĐỀ THI

TRƯỞNG BỘ MÔN
(đã duyệt)