TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA: CÔNG NGHỆ THÔNG TIN
BỘ MÔN: CÔNG NGHỆ PHẦN MỀM

ĐỀ THI CUỐI KỲ 1 năm học 2025-2026

Tên học phần: Toán ứng dụng CNTT
Mã đề: Đ0001 Số tín chỉ: **03**
Phương pháp đánh giá (*): Tự luận có giám sát Thời gian làm bài: **90** phút

Họ tên: Nguyễn Chí Thành **Lớp**: 23T_DT2 **MSSV**: 102230268

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 (*2 điểm*): Cho dãy Fibonacci f(n)

a) Hãy viết hàm tìm số chia hết cho 7 gần nhất với f(n) nhưng không vượt quá f(n)

 # Trả lời: Mã nguồn:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

unsigned long long fib(unsigned int n) {
   if (n == 0) return 0ULL;
   if (n == 1) return 1ULL;
   unsigned long long a = 0ULL, b = 1ULL, c;
   for (unsigned int i = 2; i <= n; ++i) {
      c = a + b;

      if (c < b) return b;
      a = b;
      b = c;
   }
   return b;
}

unsigned long long nearest_multiple_of_7_leq(unsigned long long x) {
   return x - (x % 7ULL);
}

unsigned long long largest_fib_div_by_7_leq(unsigned int n) {
   unsigned long long limit = fib(n);
   unsigned long long a = 0ULL, b = 1ULL;
   unsigned long long best = (a % 7ULL == 0ULL) ? a : 0ULL;
   if (0ULL == limit) return 0ULL;

   if (1ULL <= limit && (1ULL % 7ULL == 0ULL)) best = 1ULL;
```

```
    for (unsigned int i = 2; ; ++i) {
        unsigned long long c = a + b;
        if (c < b) break;
        if (c > limit) break;
        if (c % 7ULL == 0ULL) best = c;
        a = b;
        b = c;
    }
    return best;
}

int main() {
    unsigned int n;
    printf("Nhap n (n >= 0): ");
    if (scanf("%u", &n) != 1) {
        printf("Du lieu khong hop le.\n");
        return 1;
    }

    unsigned long long fn = fib(n);
    printf("f(%u) = %llu\n", n, fn);

    unsigned long long nearest7 = nearest_multiple_of_7_leq(fn);
    printf("So chia het cho 7 gan nhat va <= f(%u): %llu\n", n, nearest7);

    unsigned long long fib7 = largest_fib_div_by_7_leq(n);
    if (fib7 == 0ULL) {

        if (fn == 0ULL) printf("Trong day Fibonacci, f(%u)=0 (chia het cho 7).\n", n);
        else printf("Khong co so Fibonacci (>0) nao chia het cho 7 va <= f(%u).\n", n);
    } else {
        printf("So trong day Fibonacci chia het cho 7 va gan nhat <= f(%u): %llu\n", n, fib7);
    }

    return 0;
}
```

Trả lời: Dán kết quả với **n = 200**

```
Nhap n (n >= 0): 200
f(200) = 12200160415121876738
So chia het cho 7 gan nhat va <= f(200): 12200160415121876736
So trong day Fibonacci chia het cho 7 va gan nhat <= f(200): 1100087778366101931
PS D:\Code\cpp\ToanUngDungCNTT\KTCK>
```

b) Hãy nhập hai số nguyên n, m và viết hàm $g(n,m) = f(n)^{f(m)} \bmod m$

 # Trả lời: Mã nguồn hàm

```
#include <stdio.h>
#include <stdlib.h>
```

```
#include <stdint.h>
#include <inttypes.h>
#include <math.h>

static void fib_mod_pair(unsigned long long n, unsigned long long mod, unsigned long long *fn, unsigned long long *fn1) {
    if (mod == 1ULL) {
        *fn = 0ULL; *fn1 = 0ULL; return;
    }
    unsigned long long a = 0ULL, b = 1ULL;
    unsigned long long res_f = 0ULL, res_f1 = 1ULL;


    unsigned long long mask = 1ULL;
    while ((mask << 1) && (mask << 1) <= n) mask <<= 1;

    unsigned long long cur_f = 0ULL, cur_f1 = 1ULL;

    for (; mask; mask >>= 1) {

        unsigned long long f = cur_f;
        unsigned long long g = cur_f1;


        unsigned long long t1 = (2 * g % mod + mod - f % mod) % mod;
        unsigned long long f2k = ( (__uint128_t)(f % mod) * t1 ) % mod;

        unsigned long long f2k1 = ( ( (__uint128_t)(f % mod) * (f % mod) ) + ( (__uint128_t)(g % mod) * (g % mod) ) ) %
mod;

        cur_f = f2k;
        cur_f1 = f2k1;

        if (n & mask) {

            unsigned long long next_f = cur_f1 % mod;
            unsigned long long next_f1 = ( (cur_f + cur_f1) % mod );
            cur_f = next_f;
            cur_f1 = next_f1;
        }
    }

    *fn = cur_f % mod;
    *fn1 = cur_f1 % mod;
}

static unsigned long long fib_mod(unsigned long long n, unsigned long long mod) {
    unsigned long long a, b;
    fib_mod_pair(n, mod, &a, &b);
    return a;
}

static unsigned long long powmod(unsigned long long a, unsigned long long e, unsigned long long mod) {
    if (mod == 1ULL) return 0ULL;
    unsigned long long res = 1ULL % mod;
    unsigned long long base = a % mod;
    while (e) {
        if (e & 1ULL) res = (unsigned long long)((__uint128_t)res * base % mod);
        base = (unsigned long long)((__uint128_t)base * base % mod);
        e >>= 1ULL;
    }
    return res;
}

static unsigned long long gcd_u64(unsigned long long a, unsigned long long b) {
```

```
    while (b) {
        unsigned long long t = a % b; a = b; b = t;
    }
    return a;
}

static unsigned long long phi(unsigned long long n) {
    if (n == 0ULL) return 0ULL;
    unsigned long long result = n;
    unsigned long long temp = n;
    for (unsigned long long p = 2ULL; p * p <= temp; ++p) {
        if (temp % p == 0ULL) {
            while (temp % p == 0ULL) temp /= p;
            result = result / p * (p - 1ULL);
        }
    }
    if (temp > 1ULL) result = result / temp * (temp - 1ULL);
    return result;
}

static int fib_ge_threshold(unsigned long long n, unsigned long long threshold) {
    if (threshold == 0ULL) return 1;
    unsigned long long a = 0ULL, b = 1ULL;
    if (n == 0ULL) return (0ULL >= threshold) ? 1 : 0;
    if (a >= threshold || b >= threshold) return 1;
    for (unsigned long long i = 2ULL; i <= n; ++i) {
        unsigned long long c = a + b;
        if (c < b) return 1;
        if (c >= threshold) return 1;
        a = b; b = c;
    }
    return 0;
}


static unsigned long long g_fn_pow_fm_mod_m(unsigned long long n, unsigned long long m) {
    if (m == 0ULL) {

        return 0ULL;
    }
    if (m == 1ULL) return 0ULL;

    unsigned long long base = fib_mod(n, m);


    unsigned long long ph = phi(m);


    unsigned long long exp_mod = 0ULL;
    if (ph == 0ULL) {

        exp_mod = 0ULL;
    } else {
        exp_mod = fib_mod(m, ph);
    }


    int exp_ge = 0;
    if (ph > 0ULL) exp_ge = fib_ge_threshold(m, ph);

    unsigned long long exponent_for_pow = exp_mod;
    if (exp_ge) exponent_for_pow += ph;
```

```
    if (exponent_for_pow == 0ULL) return 1ULL % m;

    return powmod(base, exponent_for_pow, m);
}

int main() {
    unsigned long long n, m;
    printf("Nhap hai so nguyen n m (m>0): ");
    if (scanf("%" SCNu64 " %" SCNu64, &n, &m) != 2) {
        fprintf(stderr, "Nhap khong hop le\n");
        return 1;
    }

    if (m == 0ULL) {
        fprintf(stderr, "m phai > 0\n");
        return 1;
    }

    unsigned long long result = g_fn_pow_fm_mod_m(n, m);
    printf("g(%" PRIu64 ", %" PRIu64 ") = %" PRIu64 "\n", n, m, result);
    return 0;
}
```

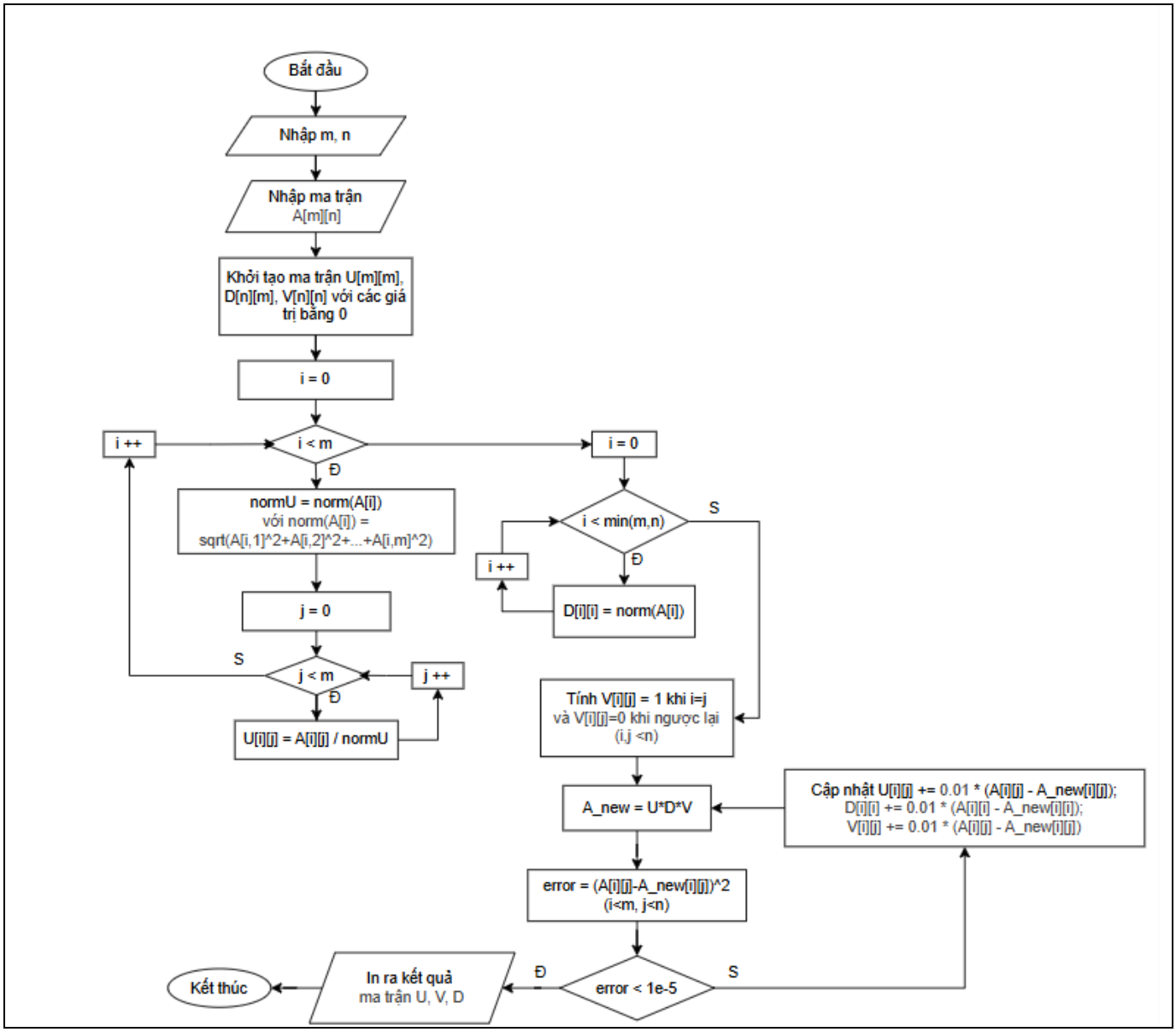
 # Trả lời: Dán kết quả với **n = 200**, **m = 97**

```
Nhap hai so nguyen n m (m>0): 200 97
g(200, 97) = 3
```

Câu 2 **(2 *điểm*): Cho ma trận A**

a) Trình bày thuật toán phân rã ma trận trận A bằng SVD sử dụng sơ đồ khối

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



b) Viết chương trình phân rã ma trận A theo SVD

 # Trả lời: Mã nguồn hàm phân rã SVD vào bên dưới:

```
#include <iostream>
#include <Eigen/Dense>
#include <iomanip>
#include <cmath>
#define eps 1e-5
using namespace std;
using namespace Eigen;

void input(double A[][10], int row, int col);
void swap(double &a, double &b);
void display(double A[][10], int row, int col);
void transpose(double A[][10], double At[][10], int rows, int cols);
void NhanMaTran(MatrixXd &S, double A[][10], double B[][10], int row1, int col1, int col2);
void GetEigenValuesAndVector(MatrixXd S, MatrixXd &lambda, MatrixXd &vector);
void tinhMatranS(MatrixXd lambda, double sigma[][10], int rows, int cols);
void tinhMatranU(MatrixXd lambda, MatrixXd vector, double U[][10], double A[][10], int rows, int
cols);
void tinhMatranV(MatrixXd vector, double V[][10]);
```

```
void gramSchmidt(double U[][10], int rows, int cols);
void computeApproximateMatrix(double A3[][10], double U[][10], double sigma[][10], double
VT[][10], int rows, int cols, int k);
double frobeniusNorm(double A[][10], double B[][10], int rows, int cols);

int main()
{
   int rows, cols;
   double A[10][10], At[10][10];
   cout << "Nhap ma tran " << endl;
   cout << "Hang: ";
   cin >> rows;
   cout << "Cot: ";
   cin >> cols;
   cout << "Nhap phan tu ma tran: " << endl;
   input(A, rows, cols);
   cout << "Ma tran goc" << endl;
   display(A, rows, cols);
   MatrixXd S(cols, cols), lambda(cols, 1), vector(cols, cols);
   transpose(A, At, rows, cols);
   NhanMaTran(S, At, A, cols, rows, cols);
   GetEigenValuesAndVector(S, lambda, vector);
   cout << "Phan ra theo U*sigma*V^T" << endl;
   double sigma[10][10], U[10][10], V[10][10];
   tinhMatranU(lambda, vector, U, A, rows, cols);
   cout << "Ma tran U:" << endl;
   display(U, rows, rows);
   tinhMatranS(lambda, sigma, rows, cols);
   cout << "Ma tran sigma:" << endl;
   display(sigma, rows, cols);
   tinhMatranV(vector, V);
   cout << "Ma tran V^T:" << endl;
   double VT[10][10];
   transpose(V, VT, cols, cols);
   display(VT, cols, cols);
   // Thử lại U * sigma * V^T - A
   MatrixXd Umat(rows, rows), Sig(rows, cols), VTrans(cols, cols);

  cout << "Kiem tra U * Sigma * V^T - A:" << endl;
  for (int i = 0; i < rows; i++)
      for (int j = 0; j < rows; j++)
         Umat(i, j) = U[i][j];

   for (int i = 0; i < rows; i++)
      for (int j = 0; j < cols; j++)
         Sig(i, j) = sigma[i][j];

   for (int i = 0; i < cols; i++)
      for (int j = 0; j < cols; j++)
         VTrans(i, j) = VT[i][j];

   MatrixXd Acheck(rows, cols);
   for (int i = 0; i < rows; i++)
```

```
    for (int j = 0; j < cols; j++)
        Acheck(i, j) = A[i][j];

MatrixXd diff = Umat * Sig * VTrans - Acheck;

cout << diff << endl;

//
==========================================================================
// TÍNH MA TRẬN XẤP XỈ A3 (sử dụng 3 giá trị kỳ dị lớn nhất)
//
==========================================================================
cout << "\n=========================================" << endl;
cout << "TINH MA TRAN XAP XI A3" << endl;
cout << "=========================================" << endl;

int k = 3; // Sử dụng 3 giá trị kỳ dị lớn nhất

// Kiểm tra nếu k vượt quá số giá trị kỳ dị có sẵn
int max_k = min(rows, cols);
if (k > max_k) {
    cout << "Canh bao: Chi co " << max_k << " gia tri ky di, su dung k = " << max_k << endl;
    k = max_k;
}

double A3[10][10];
computeApproximateMatrix(A3, U, sigma, VT, rows, cols, k);

cout << "\nMa tran xap xi A" << k << " (su dung " << k << " gia tri ky di lon nhat):" << endl;
display(A3, rows, cols);

//
==========================================================================
// TÍNH SAI SỐ FROBENIUS: ||A - A3||
//
==========================================================================
cout << "\n=========================================" << endl;
cout << "TINH SAI SO FROBENIUS" << endl;
cout << "=========================================" << endl;

double error = frobeniusNorm(A, A3, rows, cols);

cout << "\nSai so Frobenius ||A - A" << k << "||: " << fixed << setprecision(10) << error << endl;

// Tính và hiển thị ma trận sai số (A - A3)
cout << "\nMa tran sai so (A - A" << k << "):" << endl;
double diff_matrix[10][10];
for (int i = 0; i < rows; i++) {
    for (int j = 0; j < cols; j++) {
        diff_matrix[i][j] = A[i][j] - A3[i][j];
    }
}
display(diff_matrix, rows, cols);
```

```
    cout << "\n=======================================" << endl;

    return 0;
}

void input(double A[][10], int row, int col)
{
    for (int i = 0; i < row; i++)
        for (int j = 0; j < col; j++)
        {
          // cout << "a[" << i + 1 << "][" << j + 1 << "] = ";
           cin >> A[i][j];
        }
}

void display(double A[][10], int row, int col)
{
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < col; j++)
            cout << setw(9) << fixed << setprecision(5) << A[i][j];
        cout << endl;
    }
}

void swap(double &a, double &b)
{
    double temp = a;
    a = b;
    b = temp;
}

void transpose(double A[][10], double At[][10], int rows, int cols)
{
    for (int i = 0; i < rows; i++)
        for (int j = 0; j < cols; j++)
        {
            At[j][i] = A[i][j];
        }
}

void NhanMaTran(MatrixXd &S, double A[][10], double B[][10], int row1, int col1, int col2)
{
    for (int i = 0; i < row1; i++)
    {
        for (int j = 0; j < col2; j++)
        {
            S(i, j) = 0;
            for (int k = 0; k < col1; k++)
            {
                S(i, j) = S(i, j) + A[i][k] * B[k][j];
            }
```

```
        }
    }
}

void GetEigenValuesAndVector(MatrixXd S, MatrixXd &lambda, MatrixXd &vector)
{
    SelfAdjointEigenSolver<Eigen::MatrixXd> eigensolver(S);
    vector = eigensolver.eigenvectors();
    lambda = eigensolver.eigenvalues();
    int k = lambda.rows();
    int l = vector.rows();
    for (int i = 0; i < k; i++)
    {
        if (lambda(i, 0) < eps)
            lambda(i, 0) = 0;
    }
    for (int i = 0; i < k; i++)
        for (int j = i + 1; j < k; j++)
        {
            if (lambda(j, 0) > lambda(i, 0))
            {
                swap(lambda(j, 0), lambda(i, 0));
                for (int h = 0; h < l; h++)
                    swap(vector(h, i), vector(h, j));
            }
        }
}

void tinhMatranU(MatrixXd lambda, MatrixXd vector, double U[][10], double A[][10], int rows, int
cols)
{
    MatrixXd ui(rows, 1);
    double Vi[cols][10];
    for (int i = 0; i < cols; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            Vi[j][0] = vector(j, i);
        }
        NhanMaTran(ui, A, Vi, rows, cols, 1);
        for (int k = 0; k < rows; k++)
        {
            if (lambda(i, 0) != 0)
            {
                U[k][i] = (1 / sqrt(lambda(i, 0))) * ui(k, 0);
            }
            else
            {
                U[k][i] = 0;
            }
        }
    }
    if (rows > cols)
```

```
    {
        if (rows > cols)
        {
            MatrixXd U_matrix(rows, cols);
            for (int i = 0; i < rows; i++)
            {
                for (int j = 0; j < cols; j++)
                {
                    U_matrix(i, j) = U[i][j];
                }
            }
            MatrixXd orthogonal_basis = U_matrix.householderQr().householderQ();
            for (int j = cols; j < rows; j++)
            {
                for (int i = 0; i < rows; i++)
                {
                    U[i][j] = orthogonal_basis(i, j);
                }
            }
        }
    }
    else
    {
        gramSchmidt(U, rows, cols);
    }
}

void gramSchmidt(double U[][10], int rows, int cols)
{
    for (int i = 0; i < cols; i++)
    {
        for (int j = 0; j < i; j++)
        {
            double dot_product = 0;
            for (int k = 0; k < rows; k++)
            {
                dot_product += U[k][i] * U[k][j];
            }
            for (int k = 0; k < rows; k++)
            {
                U[k][i] -= dot_product * U[k][j];
            }
        }
        double norm = 0;
        for (int k = 0; k < rows; k++)
        {
            norm += U[k][i] * U[k][i];
        }
        norm = sqrt(norm);
        if (norm > 1e-10)
        {
            for (int k = 0; k < rows; k++)
            {
```

```
            U[k][i] /= norm;
        }
    }
    else
    {
        MatrixXd random_vector = MatrixXd::Random(rows, 1);
        for (int j = 0; j < i; j++)
        {
            double dot_product = 0;
            for (int k = 0; k < rows; k++)
            {
                dot_product += random_vector(k, 0) * U[k][j];
            }
            for (int k = 0; k < rows; k++)
            {
                random_vector(k, 0) -= dot_product * U[k][j];
            }
        }
        double random_norm = random_vector.norm();
        for (int k = 0; k < rows; k++)
        {
            U[k][i] = random_vector(k, 0) / random_norm;
        }
    }
  }
}

void tinhMatranV(MatrixXd vector, double V[][10])
{
  int row = vector.rows(), col = vector.cols();
  for (int i = 0; i < row; i++)
  {
    for (int j = 0; j < col; j++)
    {
        V[i][j] = vector(i, j);
    }
  }
}
void tinhMatranS(MatrixXd lambda, double sigma[][10], int rows, int cols)
{
  int k = 0;
  for (int i = 0; i < rows; i++)
    for (int j = 0; j < cols; j++)
    {
        sigma[i][j] = (i != j) ? 0 : sqrt(lambda(k, 0));
        if (i == j)
            k++;
    }
}

void computeApproximateMatrix(double A3[][10], double U[][10], double sigma[][10], double
VT[][10], int rows, int cols, int k)
{
```

```
    // Khởi tạo ma trận kết quả về 0
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            A3[i][j] = 0.0;
        }
    }

    // Tính A3 = sum_{i=1}^{k} sigma_i * u_i * v_i^T
    // Với u_i là cột thứ i của U, v_i^T là hàng thứ i của V^T
    for (int rank = 0; rank < k; rank++) {
        double sigma_val = sigma[rank][rank]; // Giá trị kỳ dị thứ rank

        // Tính tích ngoài: sigma_val * u_rank * v_rank^T
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                A3[i][j] += sigma_val * U[i][rank] * VT[rank][j];
            }
        }
    }
}

/
double frobeniusNorm(double A[][10], double B[][10], int rows, int cols)
{
    double sum = 0.0;

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            double diff = A[i][j] - B[i][j];
            sum += diff * diff;
        }
    }

    return sqrt(sum);
}
```

c) Ứng dụng SVD để nén ma trận 7x7 sau

$$A = \begin{pmatrix} 1 & 2 & 0 & 0 & \dots & 0 & 0 \\ 2 & 5 & 2 & 0 & \dots & 0 & 0 \\ 0 & 2 & 5 & 2 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 2 & 5 & 2 & 0 \\ 0 & 0 & \dots & 0 & 2 & 5 & 2 \\ 0 & 0 & \dots & 0 & 0 & 2 & 5 \end{pmatrix}$$

Trả lời: Tính toán ma trận xấp xỉ A3 :

```
Nhap ma tran
Hang: 7
Cot: 7
Nhap phan tu ma tran:
1 2 0 0 0 0 0
2 5 2 0 0 0 0
0 2 5 2 0 0 0
0 0 2 5 2 0 0
0 0 0 2 5 2 0
0 0 0 0 2 5 2
0 0 0 0 0 2 5
Ma tran goc
  1.00000  2.00000  0.00000  0.00000  0.00000  0.00000  0.00000
  2.00000  5.00000  2.00000  0.00000  0.00000  0.00000  0.00000
  0.00000  2.00000  5.00000  2.00000  0.00000  0.00000  0.00000
  0.00000  0.00000  2.00000  5.00000  2.00000  0.00000  0.00000
  0.00000  0.00000  0.00000  2.00000  5.00000  2.00000  0.00000
  0.00000  0.00000  0.00000  0.00000  2.00000  5.00000  2.00000
  0.00000  0.00000  0.00000  0.00000  0.00000  2.00000  5.00000
Phan ra theo U*sigma*V^T
Ma tran U:
  0.07387 -0.14345  0.20405 -0.24941 -0.26736 -0.21964 -0.86626
  0.28212 -0.47454  0.52143 -0.42221 -0.23339 -0.05525  0.43307
  0.43935 -0.47725  0.08557  0.37911  0.53040  0.31624 -0.21639
  0.51712 -0.14971 -0.47390  0.30576 -0.36439 -0.49768  0.10788
  0.50138  0.28143 -0.34879 -0.47303 -0.11972  0.55392 -0.05331
  0.39496  0.51782  0.28017 -0.16045  0.49932 -0.47082  0.02539
  0.21711  0.39588  0.50441  0.52232 -0.44304  0.26928 -0.01015
Ma tran sigma:
  8.63830  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
  0.00000  7.61602  0.00000  0.00000  0.00000  0.00000  0.00000
  0.00000  0.00000  6.11087  0.00000  0.00000  0.00000  0.00000
  0.00000  0.00000  0.00000  4.38563  0.00000  0.00000  0.00000
  0.00000  0.00000  0.00000  0.00000  2.74592  0.00000  0.00000
  0.00000  0.00000  0.00000  0.00000  0.00000  1.50312  0.00000
  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
```

```
Ma tran V^T:
  0.07387  0.28212  0.43935  0.51712  0.50138  0.39496  0.21711
 -0.14345 -0.47454 -0.47725 -0.14971  0.28143  0.51782  0.39588
  0.20405  0.52143  0.08557 -0.47390 -0.34879  0.28017  0.50441
 -0.24941 -0.42221  0.37911  0.30576 -0.47303 -0.16045  0.52232
 -0.26736 -0.23339  0.53040 -0.36439 -0.11972  0.49932 -0.44304
 -0.21964 -0.05525  0.31624 -0.49768  0.55392 -0.47082  0.26928
  0.86626 -0.43307  0.21639 -0.10788  0.05331 -0.02539  0.01015
Kiem tra U * Sigma * V^T - A:
-0.00010  0.00005 -0.00003  0.00001 -0.00001  0.00000 -0.00000
 0.00005 -0.00003  0.00001 -0.00001  0.00000 -0.00000  0.00000
-0.00003  0.00001 -0.00001  0.00000 -0.00000  0.00000 -0.00000
 0.00001 -0.00001  0.00000 -0.00000  0.00000 -0.00000  0.00000
-0.00001  0.00000 -0.00000  0.00000 -0.00000  0.00000 -0.00000
 0.00000 -0.00000  0.00000 -0.00000  0.00000 -0.00000  0.00000
-0.00000  0.00000 -0.00000  0.00000 -0.00000  0.00000 -0.00000
PS D:\Code\cpp\ToanUngDungCNTT\KTCK\2_SVD-eigen-cholesky>
```

```
========================================
TINH MA TRAN XAP XI A3
========================================

Ma tran xap xi A3 (su dung 3 gia tri ky di lon nhat):
  0.45829  1.34864  0.90846 -0.09736 -0.42244  0.03564  0.33498
  1.34864  4.06401  3.06819  0.29128 -0.90662 -0.01619  0.70559
  0.90846  3.06819  3.44684  2.25893  0.69754 -0.23666 -0.35117
 -0.09736  0.29128  2.25893  3.85309  2.92889  0.36256 -0.94225
 -0.42244 -0.90662  0.69754  2.92889  3.51812  2.22330  0.71373
  0.03564 -0.01619 -0.23666  0.36256  2.22330  3.86929  3.16555
  0.33498  0.70559 -0.35117 -0.94225  0.71373  3.16555  3.15556
```

Trả lời: Tính sai số giữa ma trận A và ma trận xấp xỉ A3 sử dụng độ đo Frobenius: ||A- A3||:

```
=====================================
TINH SAI SO FROBENIUS
=====================================

Sai so Frobenius ||A - A3||: 5.3882472231

Ma tran sai so (A - A3):
  0.54171  0.65136 -0.90846  0.09736  0.42244 -0.03564 -0.33498
  0.65136  0.93599 -1.06819 -0.29128  0.90662  0.01619 -0.70559
 -0.90846 -1.06819  1.55316 -0.25893 -0.69754  0.23666  0.35117
  0.09736 -0.29128 -0.25893  1.14691 -0.92889 -0.36256  0.94225
  0.42244  0.90662 -0.69754 -0.92889  1.48188 -0.22330 -0.71373
 -0.03564  0.01619  0.23666 -0.36256 -0.22330  1.13071 -1.16555
 -0.33498 -0.70559  0.35117  0.94225 -0.71373 -1.16555  1.84444
```

Câu 3 (2 *điểm*): Cho 20 điểm trong không gian 10 chiều với dữ liệu có trong tệp B.scv, hãy tìm cặp điểm có sự tương đồng cực đại

a) *Viết hàm tính độ đo tương đồng của 2 điểm sử dụng độ đo Cosine*

 # Trả lời: công thức tính độ đo:

Cho hai điểm (vector) **A** và **B** trong không gian n chiều:

$$\text{Cosine Similarity}(\mathbf{A},\mathbf{B}) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\times\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Trong đó:

- $\mathbf{A}\cdot\mathbf{B}$ là tích vô hướng của hai vector
- $\|\mathbf{A}\|$ là độ dài của vector **A**: $\|\mathbf{A}\| = \sqrt{\sum_{i=1}^{n} A_i^2}$
- $\|\mathbf{B}\|$ là độ dài của vector **B**: $\|\mathbf{B}\| = \sqrt{\sum_{i=1}^{n} B_i^2}$
- n là số chiều của không gian (ở đây n = 10)

 # Trả lời: Mã nguồn hàm tính độ đo:

```
struct Point {
   int id;
   double coords[DIMENSIONS];  // Tọa độ 10 chiều
   double norm;                // Độ dài (norm) của vector

   Point() : id(-1), norm(0.0) {
      for (int i = 0; i < DIMENSIONS; i++) {
         coords[i] = 0.0;
      }
   }
};
```

```
struct PairResult {
    int i, j;
    double similarity;

    PairResult() : i(-1), j(-1), similarity(-2.0) {}
    PairResult(int _i, int _j, double _sim) : i(_i), j(_j), similarity(_sim) {}
};

double calculateNorm(const double coords[]) {
    double sum = 0.0;
    for (int i = 0; i < DIMENSIONS; i++) {
        sum += coords[i] * coords[i];
    }
    return sqrt(sum);
}

double dotProduct(const double a[], const double b[]) {
    double sum = 0.0;
    for (int i = 0; i < DIMENSIONS; i++) {
        sum += a[i] * b[i];
    }
    return sum;
}

double cosineSimilarity(const Point& p1, const Point& p2) {
    // Tính tích vô hướng
    double dot = dotProduct(p1.coords, p2.coords);

    // Tính cosine = dot / (norm1 * norm2)
    // Tránh chia cho 0
    if (p1.norm < 1e-10 || p2.norm < 1e-10) {
        return 0.0;
    }

    return dot / (p1.norm * p2.norm);
}
```

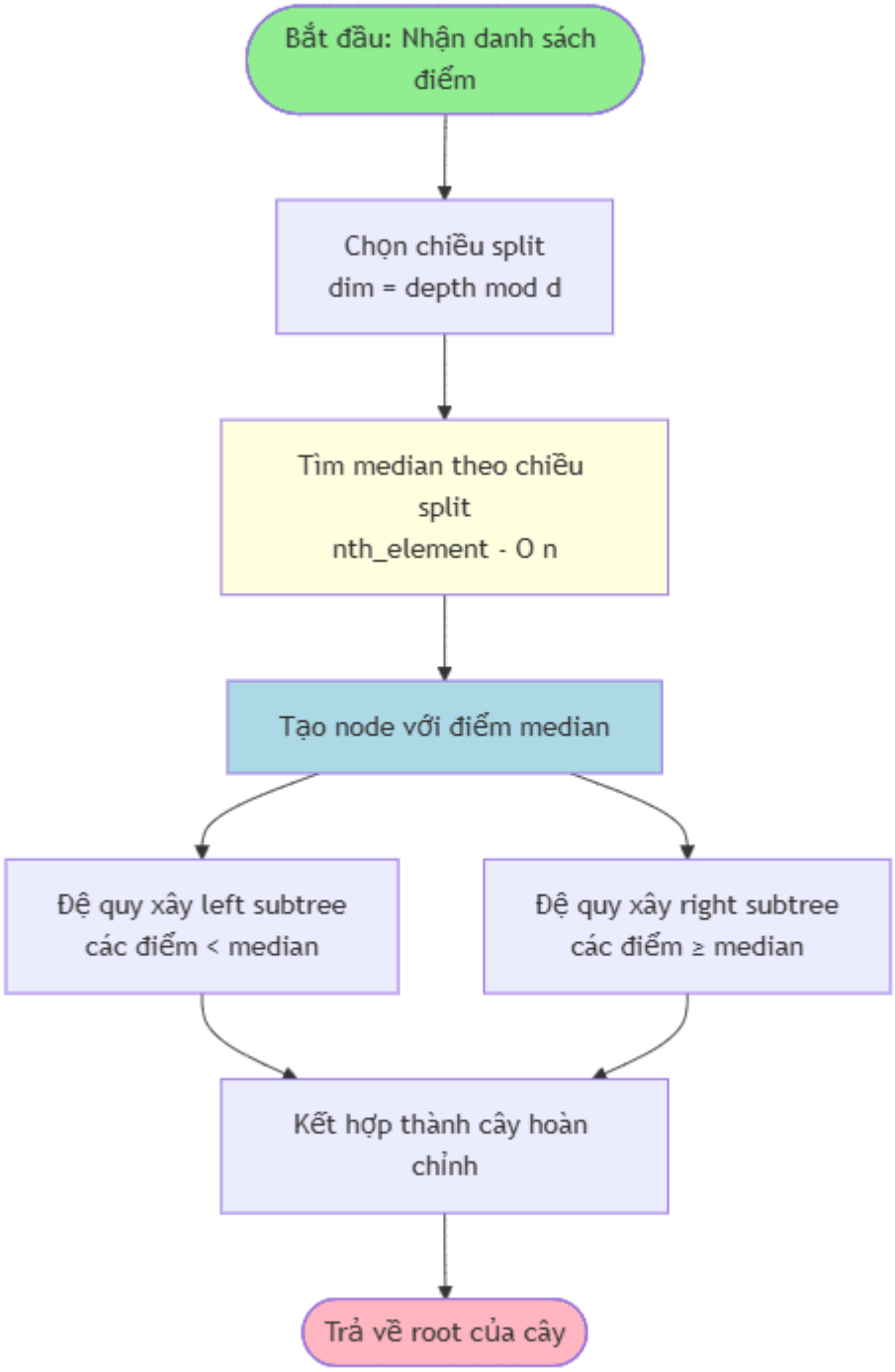
b) Xây dựng thuật toán tính độ đo tương đồng cực đại với độ phức tạp (n log n) với n là số điểm

```
# Trả lời: Trình bày thuật toán bằng mã giả/sơ đồ khối và lý giải về độ phức tạp

FUNCTION findMaxCosineSimilarity_KDTree(points[], n, d):
    // Bước 1: Chuẩn hóa - O(n×d)
    FOR i = 0 TO n-1:
        norm[i] = sqrt(sum(points[i][k]² for k in 0..d-1))
        FOR k = 0 TO d-1:
            unit[i][k] = points[i][k] / norm[i]

    // Bước 2: Xây dựng KD-Tree - O(n log n)
    root = buildKDTree(unit, 0, n-1, 0)

    // Bước 3: Tìm max similarity - O(n log n)
    max_sim = -INFINITY
```

```
    best_i = -1, best_j = -1

    FOR i = 0 TO n-1:
        j = findNearestNeighbor(root, unit[i], i)
        dist_sq = sum((unit[i][k] - unit[j][k])² for k in 0..d-1)
        cos_sim = 1.0 - 0.5 * dist_sq  // Công thức chuyển đổi

        IF cos_sim > max_sim:
            max_sim = cos_sim
            best_i = i
            best_j = j

    RETURN (best_i, best_j, max_sim)

FUNCTION buildKDTree(points[], left, right, depth):
    IF left > right: RETURN null

    dim = depth MOD d
    mid = (left + right) / 2
    nth_element(points, left, mid, right, dim)  // O(n)

    node = new KDNode(points[mid], dim)
    node.left = buildKDTree(points, left, mid-1, depth+1)
    node.right = buildKDTree(points, mid+1, right, depth+1)
    RETURN node

FUNCTION findNearestNeighbor(node, query, query_id):
    best = null
    best_dist = INFINITY

    FUNCTION search(node, depth):
        IF node is null: RETURN

        IF node.id ≠ query_id:
            dist = euclideanDistance(node.point, query)
            IF dist < best_dist:
                best_dist = dist
                best = node

        dim = depth MOD d
        diff = query[dim] - node.point[dim]

        // Chọn nhánh gần trước
        first = (diff <= 0) ? node.left : node.right
        second = (diff <= 0) ? node.right : node.left

        search(first, depth + 1)

        // Kiểm tra nhánh xa nếu có thể chứa điểm gần hơn
        IF diff² < best_dist:
            search(second, depth + 1)

    search(node, 0)
```

RETURN best.id

1. Xây dựng KD-Tree - O(n log n)



Tìm Nearest Neighbor - O(log n) average

Độ phức tạp thời gian

Bước	Độ phức tạp	Ghi chú
Đọc dữ liệu	O(n×d)	n=20, d=10
Tính norm	O(n×d)	Tính một lần
Chuẩn hóa	O(n×d)	Tính một lần

Tìm max	O(n log n × d)	Cần pruning thông minh
Tổng	**O(n log n × d)**	

Lưu ý: Với n = 20, phương pháp brute force đã đủ nhanh và đơn giản hơn.

 # Trả lời:Mã nguồn hàm thuật toán:

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <cmath>
#include <algorithm>
#include <iomanip>
#include <limits>
#include <string>
#include <functional>

using namespace std;

const int DIMENSIONS = 10;
const int NUM_POINTS = 20;

struct Point
{
   int id;
   double coords[DIMENSIONS];
   double norm;
   double unit[DIMENSIONS];

   Point() : id(-1), norm(0.0)
   {
      for (int i = 0; i < DIMENSIONS; i++)
      {
         coords[i] = 0.0;
         unit[i] = 0.0;
      }
   }
};

struct PairResult
{
   int i, j;
   double similarity;

   PairResult() : i(-1), j(-1), similarity(-2.0) {}
   PairResult(int _i, int _j, double _sim) : i(_i), j(_j), similarity(_sim) {}
};

double calculateNorm(const double coords[])
{
```

```
    double sum = 0.0;
    for (int i = 0; i < DIMENSIONS; i++)
    {
        sum += coords[i] * coords[i];
    }
    return sqrt(sum);
}

double dotProduct(const double a[], const double b[])
{
    double sum = 0.0;
    for (int i = 0; i < DIMENSIONS; i++)
    {
        sum += a[i] * b[i];
    }
    return sum;
}

double cosineSimilarity(const Point &p1, const Point &p2)
{

    double dot = dotProduct(p1.coords, p2.coords);

    if (p1.norm < 1e-10 || p2.norm < 1e-10)
    {
        return 0.0;
    }

    return dot / (p1.norm * p2.norm);
}

bool compareByDim0(const Point &a, const Point &b)
{
    return a.coords[0] < b.coords[0];
}

PairResult divideAndConquerHelper(vector<Point> &points, int left, int right)
{
    int n = right - left + 1;

    if (n <= 3)
    {
        PairResult best(-1, -1, -2.0);
        for (int i = left; i <= right - 1; i++)
        {
            for (int j = i + 1; j <= right; j++)
            {
                double sim = cosineSimilarity(points[i], points[j]);
                if (sim > best.similarity)
                {
                    best.i = i;
                    best.j = j;
                    best.similarity = sim;
```

```
            }
        }
    }
    return best;
}

int mid = (left + right) / 2;

PairResult leftBest = divideAndConquerHelper(points, left, mid);
PairResult rightBest = divideAndConquerHelper(points, mid + 1, right);

PairResult best = (leftBest.similarity > rightBest.similarity) ? leftBest : rightBest;

double midValue = points[mid].coords[0];
double threshold = 0.5;

vector<int> strip;
for (int i = left; i <= right; i++)
{
    if (fabs(points[i].coords[0] - midValue) < threshold)
    {
        strip.push_back(i);
    }
}

for (size_t i = 0; i < strip.size() - 1; i++)
{
    for (size_t j = i + 1; j < strip.size(); j++)
    {
        int idx1 = strip[i];
        int idx2 = strip[j];

        if ((idx1 <= mid && idx2 > mid) || (idx1 > mid && idx2 <= mid))
        {
            double sim = cosineSimilarity(points[idx1], points[idx2]);
            if (sim > best.similarity)
            {
                best.i = idx1;
                best.j = idx2;
                best.similarity = sim;
            }
        }
    }
}

return best;
}

PairResult findMaxSimilarityDivideConquer(vector<Point> &points)
{

int n = points.size();
vector<int> idx(n);
```

```
    for (int i = 0; i < n; ++i)
        idx[i] = i;

    struct KDNode
    {
        int index;
        int dim;
        KDNode *left;
        KDNode *right;
        KDNode(int idx = -1, int d = 0) : index(idx), dim(d), left(nullptr), right(nullptr) {}
    };

    auto cmpDim = [&](int a, int b, int dim)
    {
        return points[a].unit[dim] < points[b].unit[dim];
    };

    function<KDNode *(int, int, int)> build = [&](int l, int r, int depth) -> KDNode *
    {
        if (l > r)
            return nullptr;
        int dim = depth % DIMENSIONS;
        int m = (l + r) / 2;
        nth_element(idx.begin() + l, idx.begin() + m, idx.begin() + r + 1,
                    [&](int a, int b)
                    { return points[a].unit[dim] < points[b].unit[dim]; });
        KDNode *node = new KDNode(idx[m], dim);
        node->left = build(l, m - 1, depth + 1);
        node->right = build(m + 1, r, depth + 1);
        return node;
    };

    auto sqDistUnit = [&](int a, int b)
    {
        double s = 0.0;
        for (int k = 0; k < DIMENSIONS; ++k)
        {
            double d = points[a].unit[k] - points[b].unit[k];
            s += d * d;
        }
        return s;
    };

    function<void(KDNode *, int, int &, double &)> nearest = [&](KDNode *node, int targetIdx, int
&bestIdx, double &bestDist)
    {
        if (!node)
            return;
        int cur = node->index;
        if (cur != targetIdx)
        {
            double d = sqDistUnit(cur, targetIdx);
            if (d < bestDist)
```

```
            {
                bestDist = d;
                bestIdx = cur;
            }
        }

        int dim = node->dim;
        double diff = points[targetIdx].unit[dim] - points[cur].unit[dim];
        KDNode *first = diff <= 0 ? node->left : node->right;
        KDNode *second = diff <= 0 ? node->right : node->left;

        if (first)
            nearest(first, targetIdx, bestIdx, bestDist);
        double diff2 = diff * diff;
        if (second && diff2 < bestDist + 1e-18)
            nearest(second, targetIdx, bestIdx, bestDist);
    };

    KDNode *root = build(0, n - 1, 0);

    PairResult bestPair(-1, -1, -2.0);

    for (int i = 0; i < n; ++i)
    {
        int bestIdx = -1;
        double bestDist = numeric_limits<double>::infinity();
        nearest(root, i, bestIdx, bestDist);
        if (bestIdx != -1)
        {

            double cosine = 1.0 - 0.5 * bestDist;
            if (cosine > bestPair.similarity)
            {
                bestPair = PairResult(i, bestIdx, cosine);
            }
        }
    }

    function<void(KDNode *)> freeNode = [&](KDNode *node)
    {
        if (!node)
            return;
        freeNode(node->left);
        freeNode(node->right);
        delete node;
    };
    freeNode(root);

    return bestPair;
}

bool readDataFromCSV(const string &filename, vector<Point> &points)
{
```

```
    ifstream file(filename);
    if (!file.is_open())
    {
        cerr << "Khong the mo file: " << filename << endl;
        return false;
    }

    string line;
    int pointId = 0;

    getline(file, line);

    while (getline(file, line) && pointId < NUM_POINTS)
    {
        Point p;
        p.id = pointId;

        stringstream ss(line);
        string value;
        int dim = 0;

        while (getline(ss, value, ',') && dim < DIMENSIONS)
        {
            p.coords[dim] = stod(value);
            dim++;
        }

        p.norm = calculateNorm(p.coords);

        if (p.norm > 1e-12)
        {
            for (int k = 0; k < DIMENSIONS; ++k)
                p.unit[k] = p.coords[k] / p.norm;
        }
        else
        {
            for (int k = 0; k < DIMENSIONS; ++k)
                p.unit[k] = 0.0;
        }

        points.push_back(p);
        pointId++;
    }

    file.close();
    return true;
}

void printPoint(const Point &p)
{
    cout << "Diem " << p.id << ": [";
    for (int i = 0; i < DIMENSIONS; i++)
    {
```

```
        cout << fixed << setprecision(2) << p.coords[i];
        if (i < DIMENSIONS - 1)
            cout << ", ";
    }
    cout << "]" << endl;
    cout << "  Norm: " << fixed << setprecision(6) << p.norm << endl;
}

void printResult(const PairResult &result, const vector<Point> &points)
{

    if (result.i == -1 || result.j == -1)
    {
        cout << "Khong tim thay cap diem nao!" << endl;
        return;
    }

    cout << "\nCAP DIEM CO DO TUONG DONG CUC DAI:" << endl;
    const Point &p1 = points[result.i];
    const Point &p2 = points[result.j];

    printPoint(p1);
    cout << endl;
    printPoint(p2);

    cout << "\n----------------------------------------" << endl;
    cout << "  (Cosine Similarity):" << endl;
    cout << "  " << fixed << setprecision(10) << result.similarity << endl;
}

int main()
{


    vector<Point> points;
    string filename = "B.scv.csv";

    if (!readDataFromCSV(filename, points))
    {
        return 1;
    }

    cout << "Da doc " << points.size() << " diem tu file " << filename << endl;
    cout << "Moi diem co " << DIMENSIONS << " chieu\n"
         << endl;

    cout << "xu ly bang phuong phap Divide and Conquer (KD-Tree)..." << endl;
    vector<Point> pointsCopy = points;
    PairResult result = findMaxSimilarityDivideConquer(pointsCopy);
    printResult(result, points);

    return 0;
}
```

Trả lời: Dán kết quả thực nghiệm (cho biết độ đo và thông tin thông tin 2 điểm đó):

```
Da doc 20 diem tu file B.scv.csv
Moi diem co 10 chieu

xu ly bang phuong phap Divide and Conquer (KD-Tree)...

CAP DIEM CO DO TUONG DONG CUC DAI:
Diem 13: [0.48, 0.29, 0.87, 0.89, 0.77, 0.45, 0.58, 0.91, 0.31, 0.67]
  Norm: 2.090072

Diem 17: [0.17, 0.40, 0.47, 0.59, 0.78, 0.20, 0.29, 1.00, 0.37, 0.31]
  Norm: 1.650273

----------------------------------------
  (Cosine Similarity):
  0.9343647358
PS D:\Code\cpp\ToanUngDungCNTT\KTCK\3.Geometry>
```

Câu 4 (2 *điểm*): Cho hàm

$$f(y,z) = \begin{cases} |z - zy + \log_{10}(1 + e^{-z^2})| & \text{if } z \geq y \\ |-\mathbf{zy} + \log_{10}(e^{z^2} + 1)| & \text{if } z < y \end{cases}$$

a) Trình bày thuật toán Gradient Descent with Momentum để tối ưu hàm f(y, z)

Trả lời: Trình bày thuật toán bằng mã giả:

INPUT:
- gamma: learning rate (tốc độ học)
- alpha: momentum coefficient (hệ số momentum, thường 0.9)
- y_init, z_init: điểm khởi tạo
- epsilon: ngưỡng hội tụ
- max_iter: số vòng lặp tối đa

OUTPUT:
- (y*, z*): điểm cực tiểu gần đúng
- f(y*, z*): giá trị hàm tại điểm cực tiểu

thuật toán:

1. Khởi tạo:
   y ← y_init
   z ← z_init

```
    v_y ← 0        // velocity cho y
    v_z ← 0        // velocity cho z
    iter ← 0

 2. Lặp cho đến khi hội tụ hoặc đạt max_iter:
    2.1. Tính gradient tại điểm hiện tại:
        grad_y ← ∂f/∂y(y, z)
        grad_z ← ∂f/∂z(y, z)

    2.2. Cập nhật velocity (vận tốc) với momentum:
        v_y ← alpha * v_y - gamma * grad_y
        v_z ← alpha * v_z - gamma * grad_z

    2.3. Cập nhật tham số:
        y_new ← y + v_y
        z_new ← z + v_z

    2.4. Kiểm tra hội tụ:
        diff ← sqrt((y_new - y)² + (z_new - z)²)
        if diff < epsilon:
            return (y_new, z_new)

    2.5. Cập nhật cho vòng lặp tiếp theo:
        y ← y_new
        z ← z_new
      iter ← iter + 1
 3. Return (y, z) và f(y, z)
```

 # Trả lời: Sự hội tụ của Gradient Descent with Momentum phụ thuộc vào các yếu tố gì?
Phụ thuộc vào: gamma (learning rate), gamma (momentum coefficient), điểm khởi tạo (y0,z0), điều kiện dừng epsilon, hàm f(x) và đạo hàm f'(y,z)

b) Viết hàm tính giá trị nhỏ nhất của f(y,z) sử dụng thuật toán Gradient Descent with Momentum

 # Trả lời: dán hàm vào bên dưới:

```
#include <iostream>
#include <cmath>
#include <iomanip>
#define MAX_ITER 5000
#define EPS 1e-5
using namespace std;

double objectiveFunction(double y, double z) {
   if (z >= y) {
      double val = z - z*y + log10(1.0 + exp(-z*z));
      return fabs(val);
   } else {
      double val = -z*y + log10(exp(z*z) + 1.0);
      return fabs(val);
   }
}

double gradientY(double y, double z) {
   double inner;
```

```
    if (z >= y) {
        inner = z - z*y + log10(1.0 + exp(-z*z));
    } else {
        inner = -z*y + log10(exp(z*z) + 1.0);
    }
    double sign = (inner >= 0) ? 1.0 : -1.0;
    return sign * (-z);
}

double gradientZ(double y, double z) {
    const double ln10 = log(10.0);
    double inner, dg_dz;

    if (z >= y) {
        inner = z - z*y + log10(1.0 + exp(-z*z));
        double exp_term = exp(-z*z);
        dg_dz = 1.0 - y - (2.0*z / ln10) * exp_term / (1.0 + exp_term);
    } else {
        inner = -z*y + log10(exp(z*z) + 1.0);
        double exp_term = exp(z*z);
        dg_dz = -y + (2.0*z / ln10) * exp_term / (exp_term + 1.0);
    }

    double sign = (inner >= 0) ? 1.0 : -1.0;
    return sign * dg_dz;
}

void optimizeMomentum(double gamma, double alpha, double y_init, double z_init) {
    double y = y_init;
    double z = z_init;
    double vy = 0.0;  // Velocity for y
    double vz = 0.0;  // Velocity for z
    int iterations = 0;


    cout << "Function: f(y,z)" << endl;
    cout << "  if z >= y: |z - zy + log10(1 + e^(-z^2))|" << endl;
    cout << "  if z < y:  |-zy + log10(e^(z^2) + 1)|" << endl << endl;
    cout << "Initial point: (y=" << y << ", z=" << z << ")" << endl;
    cout << "Learning rate (gamma): " << gamma << endl;
    cout << "Momentum (alpha): " << alpha << endl;
    cout << "Convergence threshold: " << EPS << endl << endl;

    while (iterations < MAX_ITER) {
        iterations++;
        double y_prev = y;
        double z_prev = z;

        double grad_y = gradientY(y, z);
        double grad_z = gradientZ(y, z);

        vy = alpha * vy - gamma * grad_y;
        vz = alpha * vz - gamma * grad_z;
```

```
        y = y + vy;
        z = z + vz;

        if (iterations % 100 == 0 || iterations <= 10) {
            cout << "Iter " << setw(5) << iterations << ": "
                 << "y = " << setw(12) << y << ", z = " << setw(12) << z
                 << ", f(y,z) = " << setw(12) << objectiveFunction(y, z) << endl;
        }

        // Kiểm tra hội tụ
        double diff = sqrt((y - y_prev)*(y - y_prev) + (z - z_prev)*(z - z_prev));
        if (diff < EPS) {
            cout << "\n Convergence reached after " << iterations << " iterations." << endl;
            cout << "Minimum value: " << objectiveFunction(y, z) << endl;
            cout << "At point: y = " << y << ", z = " << z << endl;
            return;
        }
    }

    cout << "\n Maximum iterations reached without convergence." << endl;
    cout << "Final point: y = " << y << ", z = " << z << endl;
    cout << "Function value: " << objectiveFunction(y, z) << endl;
}

int main() {

    double gamma = 0.0001;    // Learning rate
    double alpha = 0.4;     // Momentum coefficient
    double y_init = 1.0;    // Initial y
    double z_init = 1.0;    // Initial z
    optimizeMomentum(gamma, alpha, y_init, z_init);


    return 0;
}
```

 # Trả lời: mô tả kết quả thực nghiệm với

- Điểm khởi tạo: y=0, z=0
- Vận tốc ban đầu: vy=0, vz= 0
- Tốc độ học lambda = 0.0001
- Tham số vận tốc beta= 0.4
- Số lần lặp =5000
- Dán kết quả giả định sau khi chạy chương trình nghiệm (y*, z*) và f(y*, z*)

```
Function: f(y,z)
  if z >= y: |z - zy + log10(1 + e^(-z^2))|
  if z < y:  |-zy + log10(e^(z^2) + 1)|

Initial point: (y=1, z=1)
Learning rate (gamma): 0.0001
Momentum (alpha): 0.4
Convergence threshold: 1e-05

Iter     1: y =       1.0001, z =      1.00002, f(y,z) =     0.429766
Iter     2: y =      1.00004, z =     0.999996, f(y,z) =     0.429696
Iter     3: y =     0.999916, z =     0.999949, f(y,z) =     0.136144
Iter     4: y =     0.999966, z =     0.999953, f(y,z) =     0.429607
Iter     5: y =     0.999887, z =     0.999918, f(y,z) =      0.13618
Iter     6: y =     0.999955, z =     0.999928, f(y,z) =     0.429586
Iter     7: y =     0.999882, z =     0.999895, f(y,z) =      0.13619
Iter     8: y =     0.999953, z =     0.999905, f(y,z) =     0.429576
Iter     9: y =     0.999881, z =     0.999873, f(y,z) =     0.429492
Iter    10: y =     0.999752, z =     0.999824, f(y,z) =     0.136337
Iter   100: y =     0.998415, z =     0.998432, f(y,z) =     0.137997
Iter   200: y =     0.996757, z =     0.996824, f(y,z) =     0.140023
Iter   300: y =     0.995298, z =     0.995264, f(y,z) =     0.423238
Iter   400: y =     0.993651, z =     0.993637, f(y,z) =     0.421007
Iter   500: y =     0.992065, z =     0.992016, f(y,z) =     0.418841
Iter   600: y =     0.990413, z =     0.990364, f(y,z) =     0.416601
Iter   700: y =     0.988695, z =     0.988682, f(y,z) =     0.414287
Iter   800: y =     0.987037, z =     0.987005, f(y,z) =     0.412038
Iter   900: y =     0.985194, z =     0.985262, f(y,z) =     0.154089
Iter  1000: y =     0.983548, z =     0.983565, f(y,z) =     0.156082
Iter  1100: y =     0.981799, z =     0.981826, f(y,z) =      0.15818
Iter  1200: y =     0.980151, z =     0.980105, f(y,z) =     0.402749
Iter  1300: y =     0.978289, z =     0.978308, f(y,z) =      0.16238
Iter  1400: y =     0.976574, z =     0.976542, f(y,z) =     0.397952
Iter  1500: y =     0.974708, z =      0.97472, f(y,z) =     0.166639
Iter  1600: y =     0.972794, z =      0.97287, f(y,z) =     0.168891
Iter  1700: y =     0.971122, z =      0.97108, f(y,z) =     0.390653
Iter  1800: y =     0.969199, z =     0.969202, f(y,z) =     0.173142
Iter  1900: y =     0.967289, z =     0.967316, f(y,z) =     0.175378
Iter  2000: y =     0.965524, z =     0.965459, f(y,z) =     0.383187
```

```
Iter  2100: y =     0.963432, z =     0.963494, f(y,z) =     0.179874
Iter  2200: y =     0.961507, z =     0.961564, f(y,z) =     0.182113
Iter  2300: y =     0.959747, z =     0.959669, f(y,z) =      0.37552
Iter  2400: y =     0.957663, z =     0.957666, f(y,z) =     0.186569
Iter  2500: y =     0.955642, z =     0.955668, f(y,z) =     0.188891
Iter  2600: y =     0.953719, z =     0.953685, f(y,z) =     0.367579
Iter  2700: y =     0.951591, z =     0.951628, f(y,z) =     0.193528
Iter  2800: y =     0.949671, z =     0.949618, f(y,z) =     0.362251
Iter  2900: y =     0.947476, z =     0.947514, f(y,z) =     0.198207
Iter  3000: y =     0.945499, z =     0.945459, f(y,z) =     0.356788
Iter  3100: y =     0.943373, z =     0.943347, f(y,z) =     0.354015
Iter  3200: y =     0.941204, z =     0.941209, f(y,z) =     0.205282
Iter  3300: y =     0.939173, z =     0.939095, f(y,z) =     0.348521
Iter  3400: y =     0.936827, z =     0.936876, f(y,z) =     0.210164
Iter  3500: y =     0.934643, z =      0.93469, f(y,z) =     0.212589
Iter  3600: y =      0.93246, z =     0.932489, f(y,z) =     0.215008
Iter  3700: y =      0.93026, z =     0.930268, f(y,z) =     0.217435
Iter  3800: y =     0.928013, z =     0.928019, f(y,z) =     0.219902
Iter  3900: y =     0.925836, z =     0.925774, f(y,z) =     0.331265
Iter  4000: y =     0.923459, z =     0.923457, f(y,z) =      0.32823
Iter  4100: y =     0.921219, z =     0.921164, f(y,z) =     0.325337
Iter  4200: y =     0.918852, z =     0.918819, f(y,z) =     0.322312
Iter  4300: y =      0.91641, z =     0.916438, f(y,z) =     0.232479
Iter  4400: y =     0.914154, z =     0.914094, f(y,z) =     0.316303
Iter  4500: y =     0.911748, z =     0.911691, f(y,z) =     0.313242
Iter  4600: y =     0.909148, z =     0.909218, f(y,z) =     0.240212
Iter  4700: y =     0.906716, z =     0.906777, f(y,z) =     0.242782
Iter  4800: y =     0.904452, z =     0.904365, f(y,z) =     0.303981
Iter  4900: y =     0.901861, z =     0.901847, f(y,z) =     0.300736
Iter  5000: y =     0.899381, z =     0.899343, f(y,z) =     0.297603

 Maximum iterations reached without convergence.
Final point: y = 0.899381, z = 0.899343
Function value: 0.297603
```

Câu 5 (2 *điểm*): Cho lịch sử bữa sáng của một người theo thứ tự thời gian, được gọi là chuỗi trạng thái với data = **['Pho', 'Banh my', 'Bun', 'Pho', 'Pizza', 'Bun', 'Banh my', 'Pho', 'Pho', 'Pizza', 'Banh my', 'Bun', 'Pho', 'Banh my', 'Bun', 'Pho', 'Pizza', 'Banh my', 'Pizza', 'Pho', 'Bun', 'Pho', 'Banh my']** và biết việc ăn sáng hôm nay chỉ phụ thuộc vào việc ăn sáng hôm qua mà không phụ thuộc vào việc ăn sáng trước đó:

a) Viết hàm xây dựng ma trận chuyển đổi trạng thái P từ chuỗi data

 # Trả lời: Dán code cho việc xác định ma trận chuyển đổi trạng thái P

```
#include <iostream>
#include <vector>
#include <string>
#include <map>
#include <iomanip>
using namespace std;

void buildTransitionMatrix(const vector<string>& data,
                           vector<string>& states,
                           vector<vector<double>>& P) {
```

```
    map<string, int> stateIndex;

    for (const string& item : data) {
        if (stateIndex.find(item) == stateIndex.end()) {
            stateIndex[item] = states.size();
            states.push_back(item);
        }
    }

    int n = states.size();

    vector<vector<int>> count(n, vector<int>(n, 0));

    for (int i = 0; i < data.size() - 1; i++) {
        int from = stateIndex[data[i]];
        int to = stateIndex[data[i + 1]];
        count[from][to]++;
    }

    P.resize(n, vector<double>(n, 0.0));

    for (int i = 0; i < n; i++) {
        int total = 0;
        for (int j = 0; j < n; j++) {
            total += count[i][j];
        }

        if (total > 0) {
            for (int j = 0; j < n; j++) {
                P[i][j] = (double)count[i][j] / total;
            }
        }
    }
}

// Hàm in ma trận chuyển trạng thái
void printTransitionMatrix(const vector<string>& states,
                           const vector<vector<double>>& P) {
    int n = states.size();

    cout << "MA TRAN CHUYEN TRANG THAI P (Transition Matrix)\n";
    cout << "P[i][j] = Xac suat chuyen tu trang thai i sang trang thai j\n\n";

    // In header
    cout << setw(12) << "From \\ To";
    for (const string& state : states) {
        cout << setw(12) << state;
    }
    cout << "\n";
    cout << string(12 + 12 * n, '-') << "\n";

    // In ma trận
    for (int i = 0; i < n; i++) {
```

```
        cout << setw(12) << states[i];
        for (int j = 0; j < n; j++) {
            cout << setw(12) << fixed << setprecision(3) << P[i][j];
        }
        cout << "\n";
    }
    cout << "\n";
}
void printStatistics(const vector<string>& data,
                     const vector<string>& states,
                     const vector<vector<double>>& P) {


    // Đếm tần suất xuất hiện của mỗi món
    map<string, int> frequency;
    for (const string& item : data) {
        frequency[item]++;
    }

    cout << "Tan suat xuat hien cac mon:\n";
    for (const string& state : states) {
        cout << "  " << setw(10) << state << ": "
             << frequency[state] << " lan ("
             << fixed << setprecision(2)
             << (frequency[state] * 100.0 / data.size()) << "%)\n";
    }
}


int main() {
    // Dữ liệu lịch sử ăn sáng
    vector<string> data = {
        "Pho", "Banh my", "Bun", "Pho", "Pizza",
        "Bun", "Banh my", "Pho", "Pho", "Pizza",
        "Banh my", "Bun", "Pho", "Banh my", "Bun",
         "Pho", "Pizza", "Banh my", "Pizza", "Pho",
          "Bun", "Pho", "Banh my"
    };

    cout << "PHAN TICH CHUOI MARKOV - AN SANG\n";
    // cout << "Du lieu lich su an sang:\n";
    // for (int i = 0; i < data.size(); i++) {
    //     cout << "Ngay " << setw(2) << i + 1 << ": " << data[i] << "\n";
    // }

    vector<string> states;
    vector<vector<double>> P;
    buildTransitionMatrix(data, states, P);

    printStatistics(data, states, P);

    printTransitionMatrix(states, P);
```

```
    return 0;
}
```

Trả lời: Tính toán và trình bày kết quả ma trận P(làm tròn 3 chữ số thập phân)

```
PHAN TICH CHUOI MARKOV - AN SANG
Tan suat xuat hien cac mon:
         Pho: 8 lan (34.78%)
     Banh my: 6 lan (26.09%)
         Bun: 5 lan (21.74%)
       Pizza: 4 lan (17.39%)
MA TRAN CHUYEN TRANG THAI P (Transition Matrix)
P[i][j] = Xac suat chuyen tu trang thai i sang trang thai j

   From \ To         Pho     Banh my         Bun       Pizza
------------------------------------------------------------
         Pho       0.125       0.375       0.125       0.375
     Banh my       0.200       0.000       0.600       0.200
         Bun       0.800       0.200       0.000       0.000
       Pizza       0.250       0.500       0.250       0.000
```

b) Hãy tính xác suất để hôm nay ăn ***'Banh my'*** *biết hôm qua đã ăn* ***'Pho'***

Trả lời: Dán code:

```
#include <iostream>
#include <vector>
#include <string>
#include <map>
#include <iomanip>
using namespace std;

void buildTransitionMatrix(const vector<string>& data,
                           vector<string>& states,
                           vector<vector<double>>& P) {
    map<string, int> stateIndex;

    for (const string& item : data) {
        if (stateIndex.find(item) == stateIndex.end()) {
            stateIndex[item] = states.size();
            states.push_back(item);
        }
    }

    int n = states.size();

    vector<vector<int>> count(n, vector<int>(n, 0));

    for (int i = 0; i < data.size() - 1; i++) {
```

```
        int from = stateIndex[data[i]];
        int to = stateIndex[data[i + 1]];
        count[from][to]++;
    }

    P.resize(n, vector<double>(n, 0.0));

    for (int i = 0; i < n; i++) {
        int total = 0;
        for (int j = 0; j < n; j++) {
            total += count[i][j];
        }

        if (total > 0) {
            for (int j = 0; j < n; j++) {
                P[i][j] = (double)count[i][j] / total;
            }
        }
    }
}

// Hàm in ma trận chuyển trạng thái
void printTransitionMatrix(const vector<string>& states,
                           const vector<vector<double>>& P) {
    int n = states.size();

    cout << "MA TRAN CHUYEN TRANG THAI P (Transition Matrix)\n";
    cout << "P[i][j] = Xac suat chuyen tu trang thai i sang trang thai j\n\n";

    // In header
    cout << setw(12) << "From \\ To";
    for (const string& state : states) {
        cout << setw(12) << state;
    }
    cout << "\n";
    cout << string(12 + 12 * n, '-') << "\n";

    // In ma trận
    for (int i = 0; i < n; i++) {
        cout << setw(12) << states[i];
        for (int j = 0; j < n; j++) {
            cout << setw(12) << fixed << setprecision(4) << P[i][j];
        }
        cout << "\n";
    }
    cout << "\n";
}

// Hàm tính xác suất chuyển từ một trạng thái sang trạng thái khác
double getProbability(const vector<string>& states,
                      const vector<vector<double>>& P,
                      const string& from,
                      const string& to) {
```

```
    int fromIndex = -1, toIndex = -1;

    for (int i = 0; i < states.size(); i++) {
        if (states[i] == from) fromIndex = i;
        if (states[i] == to) toIndex = i;
    }

    if (fromIndex == -1 || toIndex == -1) {
        cout << "Loi: Trang thai khong ton tai!\n";
        return -1;
    }

    return P[fromIndex][toIndex];
}

// Hàm in thống kê dữ liệu
void printStatistics(const vector<string>& data,
                     const vector<string>& states,
                     const vector<vector<double>>& P) {


    // Đếm tần suất xuất hiện của mỗi món
    map<string, int> frequency;
    for (const string& item : data) {
        frequency[item]++;
    }

    cout << "Tan suat xuat hien cac mon:\n";
    for (const string& state : states) {
        cout << "  " << setw(10) << state << ": "
             << frequency[state] << " lan ("
             << fixed << setprecision(2)
             << (frequency[state] * 100.0 / data.size()) << "%)\n";
    }
}

int main() {
    // Dữ liệu lịch sử ăn sáng
    vector<string> data = {
        "Pho", "Banh my", "Bun", "Pho", "Pizza",
        "Bun", "Banh my", "Pho", "Pho", "Pizza",
        "Banh my", "Bun", "Pho", "Banh my", "Bun",
         "Pho", "Pizza", "Banh my", "Pizza", "Pho",
          "Bun", "Pho", "Banh my"
    };

    cout << "PHAN TICH CHUOI MARKOV - AN SANG\n";

    vector<string> states;
    vector<vector<double>> P;
    buildTransitionMatrix(data, states, P);

    printStatistics(data, states, P);
```

```
    printTransitionMatrix(states, P);

    string from = "Pho";
    string to = "Banh my";

    double prob = getProbability(states, P, from, to);

    cout << "TINH XAC SUAT\n";
    cout << "Hom qua an: " << from << "\n";
    cout << "Hom nay an: " << to << "\n";
    cout << "Xac suat P(" << to << " | " << from << ") = "
         << fixed << setprecision(3) << prob << "\n";



    return 0;
}
```

Trả lời: Kết quả xác suất:

```
PHAN TICH CHUOI MARKOV - AN SANG
Tan suat xuat hien cac mon:
         Pho: 8 lan (34.78%)
     Banh my: 6 lan (26.09%)
         Bun: 5 lan (21.74%)
       Pizza: 4 lan (17.39%)
MA TRAN CHUYEN TRANG THAI P (Transition Matrix)
P[i][j] = Xac suat chuyen tu trang thai i sang trang thai j

   From \ To         Pho     Banh my         Bun       Pizza
------------------------------------------------------------
         Pho       0.125       0.375       0.125       0.375
     Banh my       0.200       0.000       0.600       0.200
         Bun       0.800       0.200       0.000       0.000
       Pizza       0.250       0.500       0.250       0.000

TINH XAC SUAT
Hom qua an: Pho
Hom nay an: Banh my
Xac suat P(Banh my | Pho) = 0.375
```

c) Tính xác suất để người đó ăn '**Pizza**' vào ba ngày nữa

Trả lời: Dán code:

```
#include <iostream>
#include <vector>
#include <string>
#include <map>
```

```
#include <iomanip>
using namespace std;

void buildTransitionMatrix(const vector<string> &data,
                           vector<string> &states,
                           vector<vector<double>> &P)
{
   map<string, int> stateIndex;

   for (const string &item : data)
   {
      if (stateIndex.find(item) == stateIndex.end())
      {
         stateIndex[item] = states.size();
         states.push_back(item);
      }
   }

   int n = states.size();

   vector<vector<int>> count(n, vector<int>(n, 0));

   for (int i = 0; i < data.size() - 1; i++)
   {
      int from = stateIndex[data[i]];
      int to = stateIndex[data[i + 1]];
      count[from][to]++;
   }

   P.resize(n, vector<double>(n, 0.0));

   for (int i = 0; i < n; i++)
   {
      int total = 0;
      for (int j = 0; j < n; j++)
      {
         total += count[i][j];
      }

      if (total > 0)
      {
         for (int j = 0; j < n; j++)
         {
            P[i][j] = (double)count[i][j] / total;
         }
      }
   }
}

void printTransitionMatrix(const vector<string> &states,
                           const vector<vector<double>> &P)
{
   int n = states.size();
```

```
    cout << "MA TRAN CHUYEN TRANG THAI P (Transition Matrix)\n";
    cout << "P[i][j] = Xac suat chuyen tu trang thai i sang trang thai j\n\n";

    cout << setw(12) << "From \\ To";
    for (const string &state : states)
    {
        cout << setw(12) << state;
    }
    cout << "\n";
    cout << string(12 + 12 * n, '-') << "\n";

    for (int i = 0; i < n; i++)
    {
        cout << setw(12) << states[i];
        for (int j = 0; j < n; j++)
        {
            cout << setw(12) << fixed << setprecision(4) << P[i][j];
        }
        cout << "\n";
    }
    cout << "\n";
}

double getProbability(const vector<string> &states,
                      const vector<vector<double>> &P,
                      const string &from,
                      const string &to)
{
    int fromIndex = -1, toIndex = -1;

    for (int i = 0; i < states.size(); i++)
    {
        if (states[i] == from)
            fromIndex = i;
        if (states[i] == to)
            toIndex = i;
    }

    if (fromIndex == -1 || toIndex == -1)
    {
        cout << "Loi: Trang thai khong ton tai!\n";
        return -1;
    }

    return P[fromIndex][toIndex];
}

void printStatistics(const vector<string> &data,
                     const vector<string> &states,
                     const vector<vector<double>> &P)
{
```

```
    map<string, int> frequency;
    for (const string &item : data)
    {
        frequency[item]++;
    }

    cout << "Tan suat xuat hien cac mon:\n";
    for (const string &state : states)
    {
        cout << "  " << setw(10) << state << ": "
             << frequency[state] << " lan ("
             << fixed << setprecision(2)
             << (frequency[state] * 100.0 / data.size()) << "%)\n";
    }
}

vector<vector<double>> multiplyMatrix(const vector<vector<double>> &A, const
vector<vector<double>> &B)
{
    int n = A.size();
    vector<vector<double>> C(n, vector<double>(n, 0.0));
    for (int i = 0; i < n; ++i)
    {
        for (int k = 0; k < n; ++k)
        {
            if (A[i][k] == 0.0)
                continue;
            for (int j = 0; j < n; ++j)
            {
                C[i][j] += A[i][k] * B[k][j];
            }
        }
    }
    return C;
}

vector<vector<double>> matrixPower(vector<vector<double>> P, int power)
{
    int n = P.size();

    vector<vector<double>> result(n, vector<double>(n, 0.0));
    for (int i = 0; i < n; ++i)
        result[i][i] = 1.0;

    while (power > 0)
    {
        if (power & 1)
            result = multiplyMatrix(result, P);
        P = multiplyMatrix(P, P);
        power >>= 1;
    }
    return result;
}
```

```
vector<double> multiplyVectorMatrix(const vector<double> &v, const vector<vector<double>> &M)
{
   int n = v.size();
   vector<double> res(n, 0.0);
   for (int j = 0; j < n; ++j)
   {
      for (int i = 0; i < n; ++i)
      {
         res[j] += v[i] * M[i][j];
      }
   }
   return res;
}

double probabilityAfterNDays(const vector<string> &states,
                             const vector<vector<double>> &P,
                             const string &start,
                             const string &target,
                             int days)
{
   int n = states.size();
   int startIdx = -1, targetIdx = -1;
   for (int i = 0; i < n; ++i)
   {
      if (states[i] == start)
         startIdx = i;
      if (states[i] == target)
         targetIdx = i;
   }
   if (startIdx == -1 || targetIdx == -1)
      return 0.0;

   if (days == 0)
      return (startIdx == targetIdx) ? 1.0 : 0.0;

   vector<vector<double>> Ppower = matrixPower(P, days);

   vector<double> v(n, 0.0);
   v[startIdx] = 1.0;

   vector<double> vAfter = multiplyVectorMatrix(v, Ppower);
   return vAfter[targetIdx];
}

int main()
{

   vector<string> data = {
      "Pho", "Banh my", "Bun", "Pho", "Pizza",
      "Bun", "Banh my", "Pho", "Pho", "Pizza",
      "Banh my", "Bun", "Pho", "Banh my", "Bun",
      "Pho", "Pizza", "Banh my", "Pizza", "Pho",
```

```
        "Bun", "Pho", "Banh my"};

    cout << "PHAN TICH CHUOI MARKOV - AN SANG\n";

    vector<string> states;
    vector<vector<double>> P;
    buildTransitionMatrix(data, states, P);

    printStatistics(data, states, P);

    printTransitionMatrix(states, P);

    string from = "Pho";
    string to = "Banh my";

    double prob = getProbability(states, P, from, to);

    cout << "TINH XAC SUAT\n";
    cout << "Hom qua an: " << from << "\n";
    cout << "Hom nay an: " << to << "\n";
    cout << "Xac suat P(" << to << " | " << from << ") = "
         << fixed << setprecision(3) << prob << "\n";

    string start = to;
    string target = "Pizza";
    int days = 3;
    double prob_after_3 = probabilityAfterNDays(states, P, start, target, days);
    cout << "Xac suat de nguoi do an '" << target << "' sau " << days << " ngay (bat dau tu '" << start
<< "'): "
         << fixed << setprecision(3) << prob_after_3 << "\n";

    return 0;
}
```

Trả lời: Kết quả xác suất:

```
  From \ To         Pho     Banh my         Bun       Pizza
------------------------------------------------------------
       Pho       0.125       0.375       0.125       0.375
    Banh my      0.200       0.000       0.600       0.200
       Bun       0.800       0.200       0.000       0.000
     Pizza       0.250       0.500       0.250       0.000

TINH XAC SUAT
Hom qua an: Pho
Hom nay an: Banh my
Xac suat P(Banh my | Pho) = 0.375
Xac suat de nguoi do an 'Pizza' sau 3 ngay (bat dau tu 'Banh my'): 0.267
PS D:\Code\cpp\ToanUngDungCNTT\KTCK\5_MarkovChain>
```

Đà Nẵng, ngày 4 tháng 12 năm 2025

GIẢNG VIÊN BIÊN SOẠN ĐỀ THI

TRƯỞNG BỘ MÔN
(đã duyệt)