|  |
| --- |
| TRƯỜNG ĐẠI HỌC BÁCH KHOA  **KHOA CÔNG NGHỆ THÔNG TIN** |

**ĐỀ THI VÀ BÀI LÀM**

Tên học phần: Toán ứng dụng CNTT

Mã học phần: Hình thức thi: *Tự luận có giám sát*

Đề số: **0003** Thời gian làm bài: 90 phút *(không kể thời gian chép/phát đề)*

Được sử dụng tài liệu khi làm bài.

**Họ tên:** ……………………………**Lớp**:……………………………**MSSV**:……………………...

Sinh viên làm bài trực tiếp trên tệp này, lưu tệp với định dạng MSSV\_HọTên.pdf và nộp bài thông qua MS Teams.

***Câu 1*** (*2 điểm*): Cho M=50 và N=500. Viết chương trình (có sử dụng hàm) thực hiện:

1. *(1.0 điểm)* Tìm số lượng các số nguyên tố nằm trong khoảng M đến N, liệt kê và tính tổng của chúng.

|  |
| --- |
| **# Trả lời: Dán code bên dưới:**  #include <iostream>  #include <vector>  #include <cmath>  using namespace std;  bool isPrime(int n) {  if (n < 2) return false;  if (n == 2) return true;  if (n % 2 == 0) return false;    for (int i = 3; i <= sqrt(n); i += 2) {  if (n % i == 0) return false;  }  return true;  }  vector<int> findPrimesInRange(int M, int N) {  vector<int> primes;  for (int i = M; i <= N; i++) {  if (isPrime(i)) {  primes.push\_back(i);  }  }  return primes;  }  long long calculateSum(const vector<int>& primes) {  long long sum = 0;  for (int prime : primes) {  sum += prime;  }  return sum;  }  void displayResults(int M, int N, const vector<int>& primes) {  cout << "\nSo luong so nguyen to: " << primes.size() << endl;    if (primes.size() > 0) {  cout << "\nDanh sach cac so nguyen to:" << endl;  for (int i = 0; i < primes.size(); i++) {  cout << primes[i];  if (i < primes.size() - 1) cout << ", ";  }  cout << endl;    long long sum = calculateSum(primes);  cout << "\nTong cac so nguyen to: " << sum << endl;  } else {  cout << "\nKhong co so nguyen to nao trong khoang nay." << endl;  }  }  int main() {  int M, N;    cout << "Nhap gia tri M (bat dau): ";  cin >> M;  cout << "Nhap gia tri N (ket thuc): ";  cin >> N;    if (M > N) {  cout << "Loi: M phai nho hon hoac bang N!" << endl;  return 1;  }    if (M < 0 || N < 0) {  cout << "Loi: M va N phai la so nguyen duong!" << endl;  return 1;  }    vector<int> primes = findPrimesInRange(M, N);    displayResults(M, N, primes);    return 0;  }  **# Trả lời: Dán kết quả thực thi vào bên dưới:** |

1. *(1.0 điểm) Tìm số nào gần với số 300 nhất* Trong các số nguyên tố vừa tìm được ở ý a)

|  |
| --- |
| **# Trả lời: Dán code bên dưới:**  //Tìm số lượng các số nguyên tố nằm trong khoảng M đến N, liệt kê và tính tổng của chúng  #include <iostream>  #include <vector>  #include <cmath>  using namespace std;  bool isPrime(int n) {  if (n < 2) return false;  if (n == 2) return true;  if (n % 2 == 0) return false;    for (int i = 3; i <= sqrt(n); i += 2) {  if (n % i == 0) return false;  }  return true;  }  vector<int> findPrimesInRange(int M, int N) {  vector<int> primes;  for (int i = M; i <= N; i++) {  if (isPrime(i)) {  primes.push\_back(i);  }  }  return primes;  }  long long calculateSum(const vector<int>& primes) {  long long sum = 0;  for (int prime : primes) {  sum += prime;  }  return sum;  }  vector<int> findClosestPrimesTo(const vector<int>& primes, int target) {  if (primes.empty()) return {};    vector<int> closestPrimes;  int minDistance = abs(primes[0] - target);    for (int prime : primes) {  int distance = abs(prime - target);  if (distance < minDistance) {  minDistance = distance;  }  }    for (int prime : primes) {  int distance = abs(prime - target);  if (distance == minDistance) {  closestPrimes.push\_back(prime);  }  }    return closestPrimes;  }  void displayResults(int M, int N, const vector<int>& primes) {  cout << "\nSo luong so nguyen to: " << primes.size() << endl;    if (primes.size() > 0) {  int target = 300;  vector<int> closestPrimes = findClosestPrimesTo(primes, target);    if (!closestPrimes.empty()) {  int distance = abs(closestPrimes[0] - target);    if (closestPrimes.size() == 1) {  cout << "\nSo nguyen to gan voi " << target << " nhat la: " << closestPrimes[0];  cout << " (khoang cach: " << distance << ")" << endl;  } else {  cout << "\nCac so nguyen to gan voi " << target << " nhat la: ";  for (int i = 0; i < closestPrimes.size(); i++) {  cout << closestPrimes[i];  if (i < closestPrimes.size() - 1) cout << ", ";  }  cout << " (khoang cach: " << distance << ")" << endl;  }  }  } else {  cout << "\nKhong co so nguyen to nao trong khoang nay." << endl;  }  }  int main() {  int M, N;    cout << "Nhap gia tri M (bat dau): ";  cin >> M;  cout << "Nhap gia tri N (ket thuc): ";  cin >> N;    if (M > N) {  cout << "Loi: M phai nho hon hoac bang N!" << endl;  return 1;  }    if (M < 0 || N < 0) {  cout << "Loi: M va N phai la so nguyen duong!" << endl;  return 1;  }    vector<int> primes = findPrimesInRange(M, N);    displayResults(M, N, primes);    return 0;  }  **# Trả lời: Dán kết quả thực thi vào bên dưới:** |

***Câu 2*** (*3 điểm*): Cho ma trận A. Viết chương trình (có sử dụng hàm) phân rã ma trận A bằng phương pháp SVD.

|  |
| --- |
| **# Trả lời:** **Dán code vào bên dưới**  #include <iostream>  #include "eigen/Eigen/Dense"  #include <iomanip>  #include <cmath>  using namespace std;  using namespace Eigen;  const double EPS = 1e-5;  int count = 0;  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 multiplyMatrices(MatrixXd& S, double A[][10], double B[][10], int row1, int col1, int col2);  void getEigenValuesAndVectors(MatrixXd S, MatrixXd& lambda, MatrixXd& vector);  void calculateMatrixS(MatrixXd lambda, double sigma[][10], int rows, int cols);  void calculateMatrixU(MatrixXd lambda, MatrixXd vector, double U[][10], double A[][10], int rows, int cols);  void calculateMatrixV(MatrixXd vector, double V[][10]);  void gramSchmidt(double U[][10], int rows, int cols);  void rref(MatrixXd& U\_matrix);  int main()  {  int rows, cols;  double A[10][10], At[10][10];  cout << "Enter Matrix Dimensions:" << endl;  cout << "Rows: ";  cin >> rows;  cout << "Columns: ";  cin >> cols;  cout << "Enter Matrix Elements:" << endl;  for (int i = 0; i < rows; i++)  {  for (int j = 0; j < cols; j++)  {  cout << "A[" << i + 1 << "][" << j + 1 << "] = ";  cin >> A[i][j];  }  }  MatrixXd S(cols, cols), lambda(cols, 1), vector(cols, cols);  transpose(A, At, rows, cols);  multiplyMatrices(S, At, A, cols, rows, cols);  MatrixXd S\_copy = S;  rref(S\_copy);  getEigenValuesAndVectors(S, lambda, vector);  double sigma[10][10], U[10][10], V[10][10];  calculateMatrixU(lambda, vector, U, A, rows, cols);  cout << "Completed Gram-Schmidt Process..." << endl;  MatrixXd null\_space(cols, 1);  for (int i = 0; i < cols; i++)  {  null\_space(i, 0) = S\_copy(i, cols - 1);  }  cout << "Matrix U:" << endl;  display(U, rows, rows);  calculateMatrixS(lambda, sigma, rows, cols);  cout << "Matrix Sigma:" << endl;  display(sigma, rows, cols);  calculateMatrixV(vector, V);  cout << "Matrix V^T:" << endl;  double VT[10][10];  transpose(V, VT, cols, cols);  display(VT, cols, cols);  MatrixXd U\_matrix(rows, rows), V\_matrix(cols, cols), S\_matrix(rows, cols);  for (int i = 0; i < rows; i++)  for (int j = 0; j < rows; j++)  U\_matrix(i, j) = U[i][j];  for (int i = 0; i < cols; i++)  for (int j = 0; j < cols; j++)  V\_matrix(i, j) = V[i][j];  for (int i = 0; i < rows; i++)  for (int j = 0; j < cols; j++)  S\_matrix(i, j) = sigma[i][j];  MatrixXd A\_matrix = U\_matrix \* S\_matrix \* V\_matrix;  cout << "Reconstructed Matrix A:" << endl;  display(A, rows, cols);  return 0;  }  void rref(MatrixXd& U\_matrix)  {  int rows = U\_matrix.rows();  int cols = U\_matrix.cols();  int lead = 0;  for (int r = 0; r < rows; ++r)  {  if (lead >= cols)  return;  int i = r;  while (U\_matrix(i, lead) == 0)  {  ++i;  if (i == rows)  {  i = r;  ++lead;  if (lead == cols)  return;  }  }  U\_matrix.row(i).swap(U\_matrix.row(r));  U\_matrix.row(r) /= U\_matrix(r, lead);  for (int i = 0; i < rows; ++i)  {  if (i != r)  {  U\_matrix.row(i) -= U\_matrix.row(r) \* U\_matrix(i, lead);  }  }  ++lead;  }  }  void swap(double& a, double& b)  {  double temp = a;  a = b;  b = temp;  }  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(12) << setprecision(4) << A[i][j];  cout << endl;  }  }  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 multiplyMatrices(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) += A[i][k] \* B[k][j];  }  }  }  }  void getEigenValuesAndVectors(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;  ::count++;  }  }  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 calculateMatrixU(MatrixXd lambda, MatrixXd vector, double U[][10], double A[][10], int rows, int cols)  {  MatrixXd ui(rows, 1);  double Vi[10][10];  for (int i = 0; i < cols; i++)  {  for (int j = 0; j < cols; j++)  {  Vi[j][0] = vector(j, i);  }  multiplyMatrices(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;  }  }  }  }  void gramSchmidt(double U[][10], int rows, int cols)  {  for (int i = cols; i < rows; i++)  {  for (int j = 0; j < rows; j++)  {  U[j][i] = (i == j) ? 1 : 0;  }  for (int k = 0; k < i; k++)  {  double dot\_product = 0;  for (int j = 0; j < rows; j++)  {  dot\_product += U[j][i] \* U[j][k];  }  for (int j = 0; j < rows; j++)  {  U[j][i] -= dot\_product \* U[j][k];  }  }  double norm = 0;  for (int j = 0; j < rows; j++)  {  norm += U[j][i] \* U[j][i];  }  norm = sqrt(norm);  for (int j = 0; j < rows; j++)  {  U[j][i] /= norm;  }  }  }  void calculateMatrixV(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 calculateMatrixS(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++)  {  if (i == j && lambda(k, 0) != 0)  {  sigma[i][j] = sqrt(lambda(k, 0));  k++;  }  else  {  sigma[i][j] = 0;  }  }  }  }  **# Trả lời: Dán kết quả thực thi vào bên dưới biết rằng** , sai số . |

***Câu 3*** (*2 điểm*): Cho mười điểm trong không gian Oxy như sau: (2, 5); (3,7); (4,3); (2,9); (6,12); (7,16); (8,3); (9,8); (10,7); (11,12)

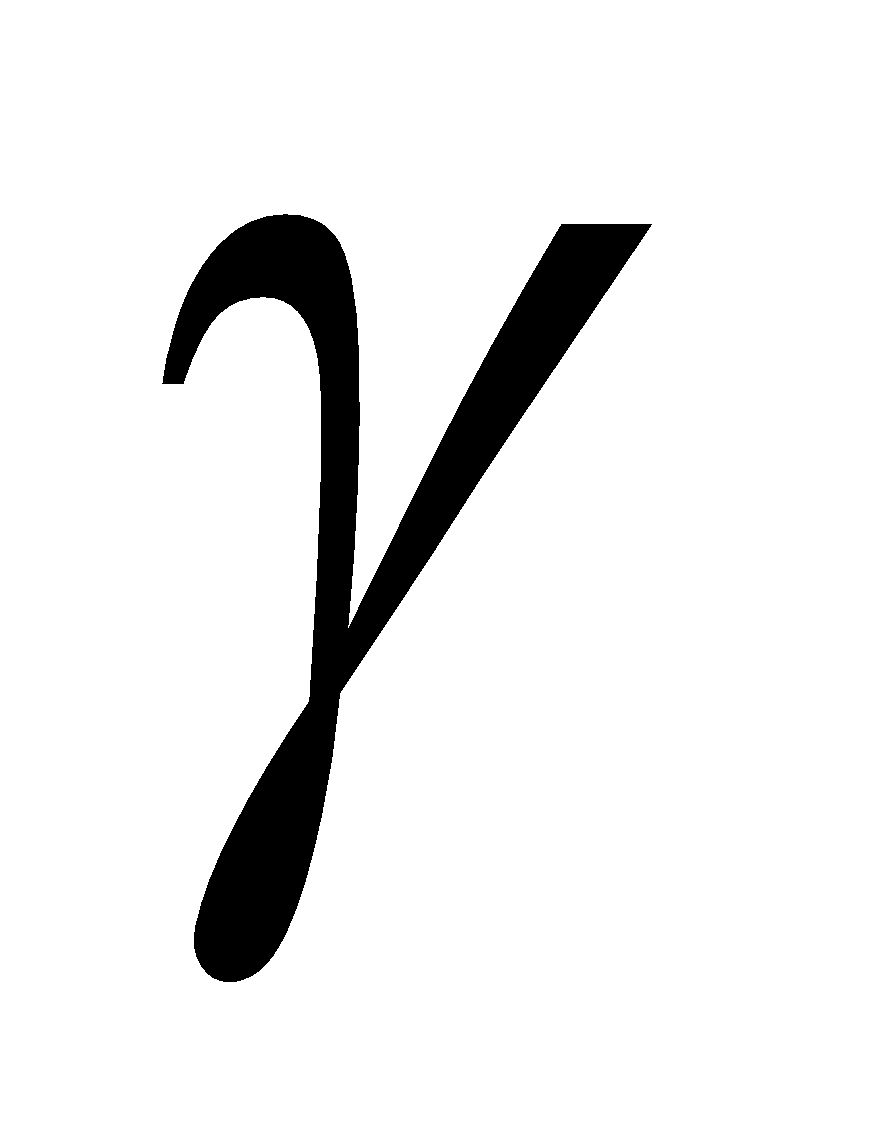
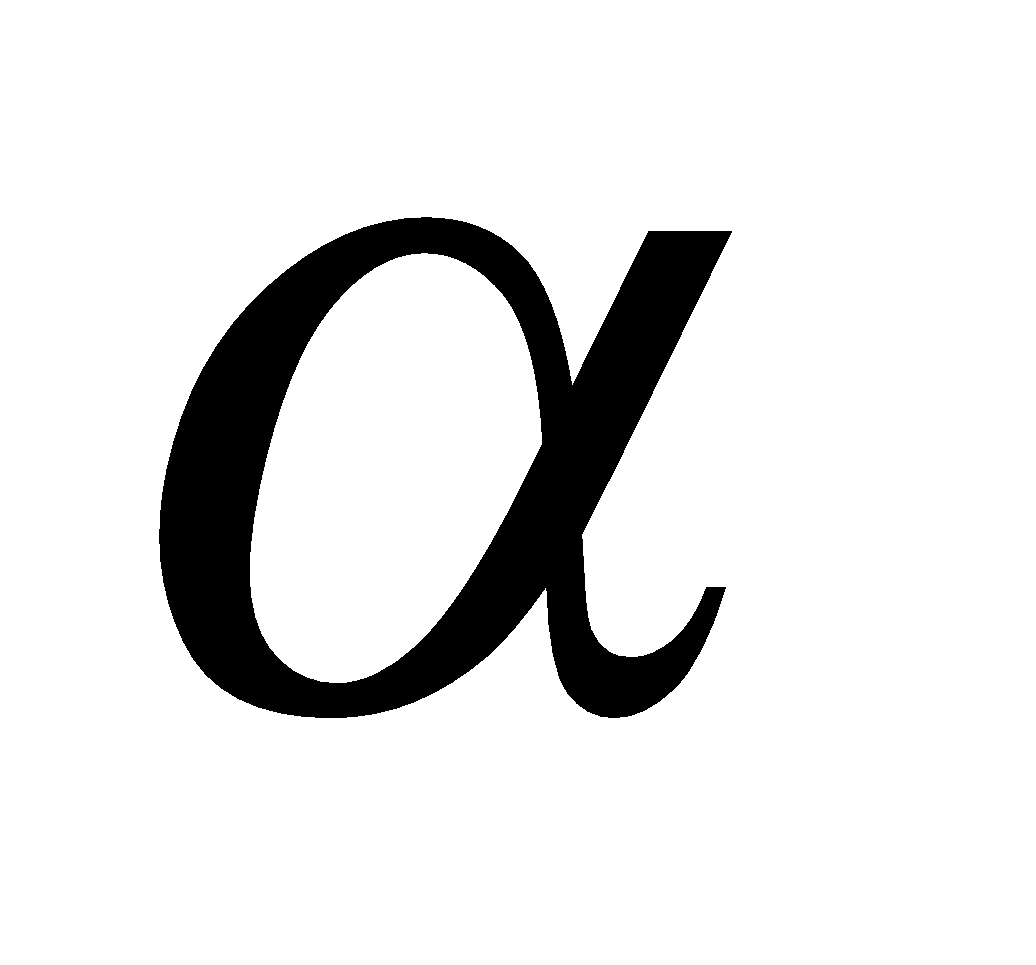
1. *(1.0 điểm) Mô tả thuật toán xác định bao lồi*

|  |
| --- |
| **# Trả lời:** **dán sơ đồ khối hoặc mã giả**  **Input**: Tập điểm P = {p1, p2,..,p3}  **Output**: Danh sách các điểm tạo thành bao lồi theo thứ tự ngược chiều kim đồng hồ  **Thuật toán GrahamScan(P):**  1. Nếu số điểm < 3, trả về P (bao lồi không xác định)  2. Tìm điểm có tung độ nhỏ nhất (nếu bằng thì hoành độ nhỏ nhất) → gọi là điểm gốc P0  3. Sắp xếp các điểm còn lại theo góc tạo bởi vector (P0 → Pi) với trục hoành (góc cực)  (nếu hai điểm có cùng góc, giữ lại điểm xa hơn P0)  4. Khởi tạo stack S, push P0, điểm đầu tiên, và điểm thứ hai sau sắp xếp vào S  5. Duyệt i từ điểm thứ ba đến hết:  while S có ít nhất 2 điểm và hướng rẽ tại 3 điểm kế tiếp không là rẽ trái:  pop điểm trên cùng khỏi S  push điểm Pi vào S  6. Stack S chứa các điểm của bao lồi theo thứ tự ngược chiều kim đồng hồ  Trả về S |

1. *(1.0 điểm)* Viết hàm xác định bao lồi và tính diện tích bao lồi tìm được.

|  |
| --- |
| **# Trả lời:** **Dán code bên dưới:**  **# Trả lời:** **Dán kết quả thực thi vào bên dưới:** |

***Câu 4*** (*2 điểm*): Cho hàm số .

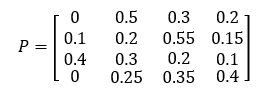
1. *(1 điểm) Trình bày thuật toán tối ưu hàm số đã cho* sử dụng phương pháp *gradient descent với momentum*, biết rằng tham số học (learning rate) , hệ số động lượng là 

|  |
| --- |
| **# Trả lời: dán sơ đồ khối hoặc mã giả** |

1. *(1 điểm)* Viết chương trình (có dùng hàm) tính giá trị bé nhất của f(x) sử dụng phương pháp *gradient descent với momentum* với số bước lặp *N* và sai số :

|  |
| --- |
| **# Trả lời**: **Dán code vào bên dưới:**  **# Trả lời**: **Dán kết quả thực thi** với điểm khởi , tham số học học (*learning rate*) , hệ số động lượng (*momentum coefficient*) là , số bước lặp và sai số : |

***Câu 5*** (*1 điểm*): Một hệ thống có chế độ làm việc ở mỗi giai đoạn vận hành chỉ với các trạng thái từ 1 đến 4. Chế độ làm việc của hệ thống này được mô tả bằng ma trận chuyển như sau:



a) (*0.5 điểm*) Vẽ đồ thị biểu diễn chuỗi Markov tương ứng đã cho

|  |
| --- |
| **# Trả lời:** **Dán kết quả vào bên dưới** |

b) (*0.5 điểm*) Giả sử rằng hệ thống bắt đầu học ở trạng thái 3. Tính xác suất hệ thống làm việc ở trạng thái 1 sau 1, 2 và 3 bước thời gian vận hành.

|  |
| --- |
| **# Trả lời**: **Dán kết quả tính toán vào bên dưới:** |

Đà Nẵng, ngày 04 tháng 12 năm 2023

|  |  |
| --- | --- |
| **GIẢNG VIÊN BIÊN SOẠN ĐỀ THI**  **Phạm Công Thắng** | **KHOA CÔNG NGHỆ THÔNG TIN**  **( đã duyệt)**  **Nguyễn Văn Hiệu** |
|  |  |