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

**ĐỀ THI GIỮA HỌC KỲ VÀ BÀI LÀM**

Tên học phần: Toán ứng dụng CNTT

Mã học phần: Hình thức thi: *Tự luận*

Đề số: **0001** Thời gian làm bài: 90 phút *(không kể thời gian chép/phát đề)*

Được sử dụng tài liệu khi làm bài.

**Họ tên:** Nguyễn Thị Hồng Diễm **Lớp**: 21T\_DT **MSSV**: 102210102

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 số nguyên dương N (N>1). Viết chương trình bằng C/C++ có sử dụng hàm thực hiện:

* Tìm các số nguyên tố nhỏ hơn N, liệt kê các số này và tính tổng của chúng, với N=25000.
* Tìm số nguyên tố gần N nhất.

|  |
| --- |
| **# Trả lời:** Dán code vào bên dưới:  **Ý 1:** Tìm các số nguyên tố nhỏ hơn N, liệt kê các số này và tính tổng của chúng, với N=25000.  #include <stdio.h>  #include <math.h>  const int MAX = 1e4+1;  //Kiem tra so nguyen to  int isPrime(int N){  if (N <= 1) return 0;  int i;  for (i = 2; i \* i <= N; i++) {  if (N % i == 0) {  return 0;  }  }  return 1;  }  long long caculatePrime(int N){  long long sum = 0;  int i;  printf("Cac so nguyen to nho hon %d la: ", N);  for(i = 1; i < N; i++){  if(isPrime(i)) {  printf("%d ", i);  sum += i;  }  }    return sum;  }  int main(){  int N;  printf("Nhap N: "); scanf("%d", &N);  printf("\nTong cac so nguyen to la: %lld ", caculatePrime(N));  }  **Ý 2:** Tìm số nguyên tố gần N nhất.  #include<stdio.h>  #include<math.h>  #define ll long long  #define MAXN 1000000  const int MAX = 1e4+1;  int prime[MAXN];  int prime[MAXN];  void sieveOfEratosthenes(){  int i,j;  for(i = 0; i <= MAXN; i++) prime[i] = 1;  prime[0] = prime[1] = 0;  for(i = 2; i\*i <= MAXN; i++){  if(prime[i]){  for(j = i\*i; j <= MAXN; j+=i) prime[j] = 0;  }  }  }  void findClosestPrimes(int n) {  int smallerPrime, largerPrime;  if (n <= 1) {  printf("2\n");  return;  }    if (prime[n]) {  printf("%d\n", n);  return;  }    int lower = n - 1;  int upper = n + 1;    while (1) {  if (prime[lower]) {  smallerPrime = lower;  break;  }  lower--;  }    while (1) {  if (prime[upper]) {  largerPrime = upper;  break;  }  upper++;  }    if(smallerPrime + largerPrime == 2\*n) printf("%d and %d\n", smallerPrime, largerPrime);  else{  smallerPrime + largerPrime > 2\*n ? printf("%d\n", smallerPrime) : printf("%d\n", largerPrime);  }  }  int main(){  int c[MAX], s[MAX];  int n;  printf("Enter n > 0: ");  scanf("%d", &n);    sieveOfEratosthenes();  printf("The closest prime number of %d: ", n); findClosestPrimes(n);  }  **# Trả lời:** Dán kết quả thực thi vào bên dưới:  **Ý 1:** Tìm các số nguyên tố nhỏ hơn N, liệt kê các số này và tính tổng của chúng, với N=25000.        **Ý 2:** Tìm số nguyên tố gần N nhất |

***Câu 2: (****2 điểm****)*** *Cho hệ phương trình đồng dư sau*



* Viết chương trình C/C++ có sử dụng hàm giải hệ phương trình đồng dư trên.

|  |
| --- |
| **# Trả lời:** Dán code vào bên dưới:  #include<stdio.h>  #include<math.h>  #define ll long long  int gcd(int a, int b){  if(b == 0) return a;  else return gcd(b, a % b);  }  int modularInverse(int x, int m){  if(gcd(x,m) != 1) return 0;  int k = 0;  while(1){  int r = 1 + k\*m;  if(r % x == 0) return r/x;  ++k;  }  }  int checkChineseRemainderTheorem(int \*a, int \*m, int n, ll s){  int x[n], x\_[n];  int i;  ll res = 0;  for(i = 0; i < n; i++){  x[i] = s/m[i];  x\_[i] = modularInverse(x[i], m[i]);  res += a[i] \* x[i] \* x\_[i];  }  return res % s;  }  int main(){  int n;  printf("Enter n > 0: "); scanf("%d", &n);  int p[n], q[n];  printf("Enter element of array a: ");  int i;  for(i = 0; i < n; i ++) scanf("%d", &p[i]);  printf("Enter element of array m: ");  for(i = 0; i < n; i ++) scanf("%d", &q[i]);    ll s = 1;  for(i = 0; i < n; i++) s \*= q[i];  printf("Chinese Remainder Theorem: x = %d + k\*%d\n",  checkChineseRemainderTheorem(p, q, n, s), s);  }  **# Trả lời:** Dán kết quả thực thi vào bên dưới: |

***Câu 3*** (*3 điểm*): Cho ma trận A. Viết chương trình bằng c/c++ có sử dụng hàm thực hiện phân rã ma trận A.

1. Phân rã **Cholesky** ma trận A

|  |
| --- |
| **# Trả lời:** Dán code vào bên dưới (bao gồm điều kiện của ma trận A nếu có):  #include<stdio.h>  #include<math.h>  #define NMAX 100  #define ll long long  //In ma tran  void printMatrix(double a[][NMAX], int n ){  int i, j, k;  for (i = 0; i < n; i++) {  for (j = 0; j < n; j++) printf("%.5lf\t", a[i][j]);  printf("\n");  }  }  //Nhap ma tran  void scanMatrix(double a[][NMAX], int n){  int i, j;  for(i = 0; i < n; i++){  for(j = 0; j < n; j++) scanf("%lf", &a[i][j]);  }  }  //Chuyen vi  void MatrixTranpose(double a[][NMAX], double tran[][NMAX], int n ){  int i, j, k;  for (i = 0; i < n; i++) {  for (j = 0; j < n; j++) tran[j][i]= a[i][j];  }  }  int checkSymmetricMatrix(double a[][NMAX], int n){  int i, j;  for(i = 0; i < n; i++){  for(j = i + 1; j < n; j++){  if(a[i][j] != a[j][i]){  return 0;  }  }  }  return 1;  }  //Tich ma tran  void multiplyMatrices(double a[][NMAX], double b[][NMAX], double res[][NMAX], int n){  int i, j, k;  for(i = 0; i < n; i++){  for(j = 0; j < n; j++){  res[i][j] = 0;  for(k = 0; k < n; k++){  res[i][j] += a[i][k] \* b[k][j];  }  }  }  }  void LDLDecomposition(double a[][NMAX], double lower[][NMAX], double D[][NMAX], int n){  int i, j, k;  for (i = 0; i < n; i++) {  for (j = 0; j <= i; j++) {  double sum = 0;  if (i == j) {  for (k = 0; k < j; k++) {  sum += lower[j][k] \* lower[j][k] \* D[k][k];  }  D[j][j] = a[j][j] - sum;  lower[j][j] = 1.0;  } else {  for (k = 0; k < j; k++) {  sum += lower[i][k] \* lower[j][k] \* D[k][k];  }  lower[i][j] =(a[i][j] - sum) / D[j][j];  }  }  }    double tran[NMAX][NMAX];  printf("\nMa tran L: \n"); printMatrix(lower, n);  printf("\nMa tran D: \n"); printMatrix(D, n);  MatrixTranpose(lower, tran, n);  printf("\nMa tran L^T:\n"); printMatrix(tran, n);  printf("\nMa tran A duoc phan ra theo Cholesky\_2 la LDL^T\n");    double res1[NMAX][NMAX], res[NMAX][NMAX];  multiplyMatrices(lower, D, res1, n);  multiplyMatrices(res1, tran, res, n);    printf("\nTich L.D.L^T: \n"); printMatrix(res,n);    int check = 0;  for(i = 0; i < n; i++){  for(j = 0; j < n; j++){  if(abs(res[i][j] - a[i][j]) > 0.01) check = 1;  }  }  if(!check) printf("\nKet qua: A = L.D.L^T => Trung voi ma tran A ban dau\n");  else printf("\nFalse: A # L.D.L^T => Khong trung voi ma tran A ban dau\n");  }  int main(){  double a[NMAX][NMAX] = {{0}};  double lower[NMAX][NMAX] = {{0}};  double transpose[NMAX][NMAX] ={{0}};  double D[NMAX][NMAX] = {{0}};  int n; printf("Enter n > 0: "); scanf("%d", &n);  printf("Enter the elements of matrix A:\n"); scanMatrix(a, n);  printf("=> A la ma tran vuong\n");  if(checkSymmetricMatrix(a, n)){  printf("=> A la ma tran doi xung\n");  LDLDecomposition(a, lower, D, n);  }  }  **# Trả lời:** Dán kết quả thực thi vào bên dưới với  (sai số ): |

1. Phân rã **eigendecomposition** ma trận A

|  |
| --- |
| **# Trả lời:** Dán code vào bên dưới (bao gồm điều kiện của ma trận A nếu có):  #include<stdio.h>  #include<math.h>  #define MAX 100  const int N = 3;  void swap(double \*a, double \*b) {  double temp = \*a;  \*a = \*b;  \*b = temp;  }  void scanMatrix(double A[MAX][MAX]){  int i, j;  for(i = 1; i <= N; i++){  for(j = 1; j <= N; j++) scanf("%lf", &A[i][j]);  }  }  void printMatrix(double A[MAX][MAX]){  int i, j;  for(i = 1; i <= N; i++){  for(j = 1; j <= N; j++) printf("%.5lf ", A[i][j]);  printf("\n");  }  }  void inverse(double mat[MAX][MAX]) {  double augmented[MAX][2\*MAX];  int i, j, k;  for (i = 1; i <= N; i++) {  for (j = 1; j <= N; j++) {  augmented[i][j] = mat[i][j];  augmented[i][j + N] = (i == j) ? 1.0 : 0.0;  }  }    for (i = 1; i <= N; i++) {  int pivot = -1;  for (j = i; j <= N; j++) {  if (augmented[j][i] != 0.0) {  pivot = j;  break;  }  }    if (pivot == -1) {  printf("Matrix is not invertible.\n");  return;  }    if (pivot != i) {  for (j = 1; j <= 2\*N; j++) {  double temp = augmented[i][j];  augmented[i][j] = augmented[pivot][j];  augmented[pivot][j] = temp;  }  }    double divisor = augmented[i][i];  for (j = 1; j <= 2\*N; j++) {  augmented[i][j] /= divisor;  }    for (j = 1; j <= N; j++) {  if (j != i) {  double factor = augmented[j][i];  for (k = 1; k <= 2\*N; k++) {  augmented[j][k] -= factor \* augmented[i][k];  }  }  }  }    for (i = 1; i <= N; i++) {  for (j = 1; j <= N; j++) {  mat[i][j] = augmented[i][j + N];  }  }  }  //Roots of Equation  void rootOfEquation(double A[MAX][MAX], double L[MAX], int\* cnt){  double a = 1;  double b = -A[1][1];  double c = -A[1][2];  double d = -A[1][3];    b /= a;  c /= a;  d /= a;    double disc, q, r, dum1, s, t, term1, r13;  q = (3.0\*c - (b\*b))/9.0;  r = -(27.0\*d) + b\*(9.0\*c - 2.0\*(b\*b));  r /= 54.0;  disc = q\*q\*q + r\*r;  term1 = (b/3.0);    double x1\_real, x2\_real, x3\_real;  double x2\_imag, x3\_imag;  if (disc > 0) // One root real, two are complex  {  s = r + sqrt(disc);  s = s<0 ? -cbrt(-s) : cbrt(s);  t = r - sqrt(disc);  t = t<0 ? -cbrt(-t) : cbrt(t);  x1\_real = -term1 + s + t;  L[1] = x1\_real;  \*cnt = 1;  }  // All roots real, at least two are equal.  else if (disc == 0){  x3\_imag = x2\_imag = 0;  r13 = r<0 ? -cbrt(-r) : cbrt(r);  x1\_real = -term1 + 2.0\*r13;  x3\_real = x2\_real = -(r13 + term1);  L[1] = x1\_real; L[2] = x2\_real; L[3] = x3\_real;  \*cnt = 3;  }  // Only option left is that all roots are real and unequal (to get here, q < 0)  else {  x3\_imag = x2\_imag = 0;  q = -q;  dum1 = q\*q\*q;  dum1 = acos(r/sqrt(dum1));  r13 = 2.0\*sqrt(q);  x1\_real = -term1 + r13\*cos(dum1/3.0);  x2\_real = -term1 + r13\*cos((dum1 + 2.0\*M\_PI)/3.0);  x3\_real = -term1 + r13\*cos((dum1 + 4.0\*M\_PI)/3.0);  L[1] = x1\_real; L[2] = x2\_real; L[3] = x3\_real;  \*cnt = 3;  }  int i;  // for(i = 1; i <= \*cnt; i++) printf("%.2lf ", L[i]);  // printf("\n");  }  //Matrix: A-lambda.I  void A\_minus\_lambdaI(double A\_minus[MAX][MAX], double A[MAX][MAX], double lambda){  int i, j;  for (i = 1; i <= N; i++) {  for (j = 1; j <= N; j++) {  A\_minus[i][j] = A[i][j] - (i == j ? lambda : 0.0);  }  }  }  //Multiply two Matrices  void multiMatrices(double a[MAX][MAX], double b[MAX][MAX], double c[MAX][MAX]) {  int i, j, k;  for (i = 1; i <= N; i++)  for (j = 1; j <= N; j++) {  c[i][j] = 0;  for (k = 1; k <= N; k++) {  c[i][j] += a[i][k]\*b[k][j];  }  }  }  //Sort desc  void sortArr(double L[MAX]){  int i, j;  for (i = 1; i <= N ; i++) {  for (j = i + 1; j <= N; j++) {  if (L[i] < L[j]) {  swap(&L[i], &L[j]);  }  }  }  }  //Matrix diagonalizable  void matrixDiagonalizable(double A[MAX][MAX]){  double L[MAX]; //gia tri rieng  double Root[MAX][MAX];  double M1[MAX][MAX], M[MAX][MAX], B[MAX][MAX], C[MAX][MAX];  int i, j, k;  int cnt;    for(i = 1; i <= N; i++) {  for(j = 1; j <= N; j++) Root[i][j] = A[i][j];  }    for(i = 1; i <= N; i++) {  for(j = 1; j <= N; j++) {  if(i == j) C[i][j] = 1;  else C[i][j] = 0;  }  }    for (k = N - 1; k >= 1; k--) {  for (i = 1; i <= N; i++) {  for (j = 1; j <= N; j++) {  if (i != k) {  if (i == j) {  M[i][j] = 1;  M1[i][j] = 1;  }  else {  M[i][j] = 0;  M1[i][j] = 0;  }  }  else {  M1[i][j] = A[k+1][j];  if(A[k+1][k] != 0){  if (j == k) M[i][j] = 1.0/A[k+1][k];  else M[i][j] = -A[k+1][j]/A[k+1][k];  }  }  }  }  multiMatrices(A, M, B);  multiMatrices(M1, B, A);  multiMatrices(C, M, B);  for(i = 1; i <= N; i++) {  for(j = 1; j <= N; j++) C[i][j] = B[i][j];  }    }      rootOfEquation(A, L, &cnt);  sortArr(L);  double P[MAX][MAX];  double R[MAX][MAX];  double K[MAX][MAX];    double diagonalMatrix[MAX][MAX];  for(i = 1; i <= N; i++) diagonalMatrix[i][i] = L[i];  printf("\nMa tran D:\n"); printMatrix(diagonalMatrix);  for(i = 1; i <= cnt; i++){  for(j = 1; j <= N; j++) K[j][1] = pow(L[i] , N - j);  multiMatrices(C, K, R);  for(j = 1; j <= N; j++) P[j][i] = R[j][1];  }    double P\_[MAX][MAX];  for(i = 1; i <= N; i++){  for(j = 1; j <= N; j++) P\_[i][j] = P[i][j];  }  inverse(P\_);    double \_P[MAX][MAX];  double Res[MAX][MAX];  multiMatrices(P, diagonalMatrix, \_P);  multiMatrices(\_P, P\_, Res);    printf("\nMa tran goc A:\n"); printMatrix(Root);  printf("\nMa tran P:\n"); printMatrix(P);  printf("\nMa tran P^-1:\n"); printMatrix(P\_);  printf("\nMa tran A sau khi phan ra theo eigendecomposition la PDP^-1\n");  printf("\nTich PDP^-1:\n"); printMatrix(Res);      int check = 0;  for(i = 1; i <= N; i++){  for(j = 1; j <= N; j++){  if(abs(Root[i][j] - Res[i][j]) > 0.00001) check = 1;  }  }  if(!check){  printf("\nKet qua: A = PDP^-1 => Trung voi ma tran ban dau");  }else printf("\nKet qua: A # PDP^-1 => Khong trung voi ma tran ban dau");    }  int main(){  double A[MAX][MAX];  printf("Nhap ma tran A:\n");  scanMatrix(A);  matrixDiagonalizable(A);  }  **# Trả lời:** Dán kết quả thực thi vào bên dưới với  (sai số ): |

***Câu 4*** (*3 điểm*): Cho ma trận A. Viết chương trình bằng c/c++ có sử dụng hàm thực hiện phân rã ma trận A bằng phương pháp SVD.

|  |
| --- |
| **# Trả lời:** Dán code vào bên dưới (bao gồm điều kiện của ma trận A nếu có):  #include <stdio.h>  #include <math.h>  #include <Eigen>  #define MAX 100  void swap(double &a, double &b){  double tmp = a;  a = b;  b = tmp;  }  void swapI(int &a, int &b){  int tmp = a;  a = b;  b = tmp;  }  //Nhap ma tran  void scanMatrix(double a[MAX][MAX], int m, int n){  for(int i = 0; i < m; i++){  for(int j = 0; j < n; j++) scanf("%lf", &a[i][j]);  }  }  //In ma tran  void printMatrix(double a[][MAX], int m, int n){  for(int i = 0; i < m; i++){  for(int j = 0; j < n; j++) printf("%.5lf\t\t", a[i][j]);  printf("\n");  }  }  //Tinh tich 2 ma tran  void multiplyMatrices(double a[][MAX], double b[][MAX], double res[][MAX], int m, int n){  for(int i = 0; i < n; i++){  for(int j = 0; j < n; j++){  res[i][j] = 0;  for(int k = 0; k < n; k++) res[i][j] += a[i][k] \* b[k][j];  }  }  }  //Tinh tich K.Matrix  void multiplyK\_Matrix(double a[][MAX], int m, int n, double k, double K[][MAX]){  int i, j;  for(i = 0; i < m; i++){  for(j = 0; j < n; j ++) K[i][j] = k\*a[i][j];  }  }  //Tim ma tran chuyen vi  void transpose(double a[][MAX], double b[][MAX], int m, int n){  for(int i = 0; i < m; i++){  for(int j = 0; j < n; j++) b[j][i] = a[i][j];  }  }  //Vector rieng tuong ung  void eigenVector(Eigen::EigenSolver<Eigen::MatrixXd> s, double v[MAX][MAX], int p, int n){  for (int i = 0; i < n; i++) {  v[i][0] = s.eigenvectors().col(p).real()(i);  }  }  void calculate(double A[MAX][MAX], double res[][MAX], int m, int n) {  Eigen::MatrixXd matrix(n, n);  for (int i = 0; i < n; i++) {  for (int j = 0; j < n; j++) matrix(i, j) = res[i][j];  }  Eigen::EigenSolver<Eigen::MatrixXd> s(matrix);    double Sigma[MAX];  int \_index[MAX];  double P[MAX][MAX];    for (int i = 0; i < n; i++) {  double eigenvalue = s.eigenvalues()(i,0).real();  if (eigenvalue > 0) Sigma[i] = sqrt(eigenvalue);  }    for(int i=0; i<n;i++) \_index[i]=i;  for(int i = 0; i < n; i++){  for(int j = 0; j < n; j ++) if(Sigma[\_index[i]] > Sigma[\_index[j]]) {  swap(Sigma[i], Sigma[j]);  swapI(\_index[i], \_index[j]);  }  }    double SigmaMatrix[MAX][MAX];  printf("\nMa tran duong cheo Sigma:\n");  int cnt = 0;  for(int i = 0; i < n; i++) \_index[i]=i;  for(int i = 0; i < n-1; i++){  for(int j = i + 1; j < n; j++){  if(Sigma[\_index[i]] < Sigma[\_index[j]]) swapI(\_index[i],\_index[j]);  }  }  for(int i=0; i < m; i++){  for(int j=0; j< n; j++){  if(i == j) SigmaMatrix[i][j]= Sigma[\_index[i]];  else SigmaMatrix[i][j]=0;  }  }  printMatrix(SigmaMatrix, m, n);    int k = 0;  double RES[MAX][MAX];  for(int i = 0; i < n; i++){  double u[MAX][MAX], U[MAX][MAX], K[MAX][MAX], v[MAX][MAX];  if(Sigma[\_index[i]] != 0){  multiplyK\_Matrix(A, m, n, 1/Sigma[\_index[i]], K);  eigenVector(s, v, \_index[i], n);  multiplyMatrices(K,v,U,m,n);  for(int q = 0; q < m; q++){  RES[q][k] = U[q][0];  }  ++k;  }  }    printf("\nMa tran U:\n"); printMatrix(RES,m,k);    double VT[MAX][MAX];  printf("\nMa tran V^T:\n");  for (int i = 0; i < n; i++) {  for (int j = 0; j < n; j++) {  VT[i][j] = s.eigenvectors()(j, \_index[i]).real();  printf("%.5lf\t\t", s.eigenvectors()(j, \_index[i]).real());  }  printf("\n");  }    double M[MAX][MAX], N[MAX][MAX];  multiplyMatrices(RES,SigmaMatrix,M,m,n);  multiplyMatrices(M, VT, N, m, n);  printf("\nMa tran U.Sigma.V^T:\n");printMatrix(N,m,n);  int check = 0;  for(int i = 1; i <= m; i++){  for(int j = 1; j <= n; j++){  if(abs(N[i][j] - A[i][j]) > 0.00001) check = 1;  }  }    printf("\nMa tran A duoc phan ra theo SVD la U.Sigma.V^T\n");    if(!check){  printf("\nKet qua: A = U.Sigma.V^T => Trung voi ma tran A ban dau\n");  }else printf("\nKet qua: A # U.Sigma.V^T => Khong trung voi ma tran A ban dau\n");    }  int main(){  int n, m;  printf("Input m and n: "); scanf("%d %d", &m, &n);  double A[MAX][MAX], B[MAX][MAX], res[MAX][MAX];  printf("Input matrix A (%d rows & %d cols):\n", m, n); scanMatrix(A, m, n);  printf("\nMatrix A:\n"); printMatrix(A, m, n);  transpose(A, B, m, n);  multiplyMatrices(B, A, res, m, n);  calculate(A, res, m, n);  return 0;  }  **# Trả lời:** Dán kết quả thực thi vào bên dưới với  (sai số ): |