TRƯỜNG ĐẠI HỌC BÁCH KHOA

**KHOA: CÔNG NGHỆ THÔNG TIN**

BỘ MÔN: CÔNG NGHỆ PHẦN MỀM



**ĐỀ THI GIỮA KỲ 1 năm học 2023-2024**

Tên học phần: Toán ứng dụng CNTT

Mã học phần: **……………………** Số tín chỉ: **03**

Phương pháp đánh giá (\*): Tự luận có giám sátThời gian làm bài: **120** phút

☐ Sinh viên không 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 MSTeam

***Câu 1*** (*2 điểm*):

1. Cho phương trình Ax + By = C với A, B, C là các số nguyên dương và x, y là biến. Hãy trình bày thuật toán để giải nghiệm nguyên của phương trình trên.

|  |
| --- |
| **# Trả lời:** Trình bày thuật toán bằng sơ đồ khối hoặc ngôn ngữ tự nhiên, nếu không có nghiệm thì trả về kết quả vô nghiệm  Giải:   * Tìm ước chung lớn nhất GCD(A,B) * Kiểm tra C có chia hết cho GCD(A,B) hay không. Nếu không thì C không phải là một bội số của GCD(A,B) nên phương trình không có nghiệm nguyên * Nếu C chia hết cho GCD(A,B) thì đặt k = C / GCD(A,B) sau đó dùng thuật toán Eculid mở rộng để tính nghiệm của phương trình Ax0 + By0 = GCD(A,B) * Sau khi tìm được x0 và y0 thì nghiệm của phương trình ban đầu là x = k.x0, y = k.y0 |

1. Hãy viết chương trình sử để giải phương trình Ax + By = C. Hãy chỉ rõ thuật toán sử dụng.

|  |
| --- |
| **# Trả lời:** Dán code vào bên dưới  Thuật toán bên dưới dùng thuật toán Eculid mở rộng đề tìm x0 và y0. Hàm gcd là hàm dùng để tìm ước chung lớn nhất  #include <iostream>  using namespace std;  long gcd(long n, long m)  {  if (n == 0) return m;  return gcd(m % n, n);  }  int extended\_gcd(int a, int b, int& x, int& y) {  if (b == 0) {  x = 1;  y = 0;  return a;  }  int x1, y1;  int d = extended\_gcd(b, a % b, x1, y1);  x = y1;  y = x1 - (a / b) \* y1;  return d;  }  void TimNghiemNguyen(long a, long b, long c)  {  long gcdNum = gcd(a, b);  if (c % gcdNum == 0)  {  int x, y;  extended\_gcd(a, b, x, y);  cout << "x = " << x \* (c / gcdNum) << " + " << b << "r, y = " << y \* (c / gcdNum) << " - " << a << "r";  }  else  {  cout << "Phuong trinh khong co nghiem nguyen";  }  }  int main()  {  long a, b, c;  cout << "Nhap a,b,c: ";  cin >> a >> b >> c;  TimNghiemNguyen(a, b, c);  }  **# Trả lời:** Dán kết quả nghiệm của phương trình 5x + 12y = 45.  A screenshot of a computer  Description automatically generated |

***Câu 2*** (3 *điểm*): Cho ma trận A

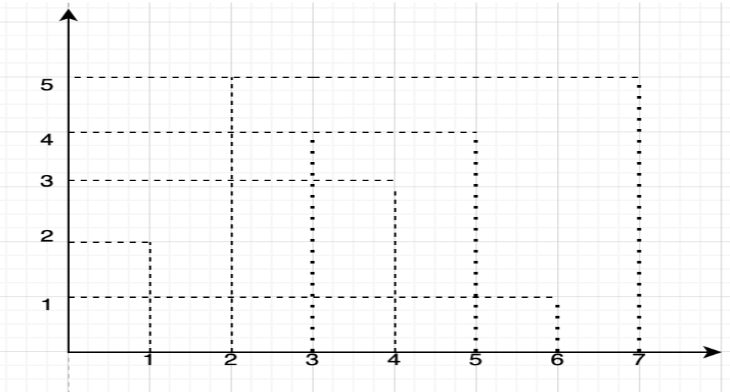
1. Trình bày thuật toán để phân rã ma trận A theo Cholesky

|  |
| --- |
| **# Trả lời:**   * Kiểm tra điều kiện của ma trận A * Ma trận vuông: Có số dòng bằng số cột * Ma trận đối xứng: Với mọi i,j thì Aij = Aji * Ma trận xác định dương: Tìm chỉ số riêng của ma trận A và nếu tất cả chỉ số riêng đều dương thì ma trận A dương. * Tiến hành xây dựng ma trận L,  trong đó:     Trong đó i = j, ... ,n-1, j = 0, ...,n-1 |

1. Viết chương trình để thực hiện phân rã Cholesky

|  |
| --- |
| **# Trả lời:** Dán code vào bên dưới:  #include <iostream>  using namespace std;  void nhap(double matrix[][100], int n)  {  for (int i = 0; i < n; i++)  {  for (int j = 0; j < n; j++)  {  cin >> matrix[i][j];  }  }  }  void printMatrix(double matrix[][100], int n)  {  for (int i = 0; i < n; i++)  {  for (int j = 0; j < n; j++)  {  cout << matrix[i][j] << " ";  }  cout << endl;  }  }  bool isSymmetricMatrix(double A[][100], int n)  {  for (int i = 0; i < n; i++)  {  for (int j = 0; j < n; j++)  {  if (A[i][j] != A[j][i])  {  return false;  }  }  }  return true;  }  bool isPositiveDefiniteMatrix(double A[][100], int n) {  for (int i = 0; i < n; i++) {  if (A[i][i] <= 0) {  return false;  }  }  for (int k = 1; k < n; k++) {  for (int i = 0; i < n - k; i++) {  double det = A[i][i] \* A[i + k][i + k] - A[i][i + k] \* A[i + k][i];  if (det <= 0) {  return false;  }  }  }  return true;  }  void choleskyDecomposition(double A[][100], double L[][100], int n)  {  for (int i = 0; i < n; i++)  {  for (int j = 0; j <= i; j++)  {  double sum = 0;  if (j == i)  {  for (int k = 0; k < j; k++)  {  sum += pow(L[j][k], 2);  }  L[j][j] = sqrt(A[j][j] - sum);  }  else  {  for (int k = 0; k < j; k++)  {  sum += L[i][k] \* L[j][k];  }  L[i][j] = (A[i][j] - sum) / L[j][j];  }  }  }  }  int main()  {  int n;  cout << "Nhap kich thuoc ma tran (n x n): ";  cin >> n;  double A[100][100];  double L[100][100] = { {0} };  nhap(A, n);  if (isSymmetricMatrix(A, n) && isPositiveDefiniteMatrix(A, n))  {  choleskyDecomposition(A, L, n);  printMatrix(L, n);  }  else  {  cout << "Ma tran khong thoa man!";  }  }  **# Trả lời:** Thực thi và dán kết quả ma trận chéo trên khi phân rã Cholesky của ma trận 10 x 10 |

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



1. *Trình bày thuật toán xác định bao lồi*

|  |
| --- |
| **# Trả lời:** dán sơ đồ khối hoặc ngôn ngữ tự nhiên vào bên dưới:  B1: Sắp xếp các điểm tăng dần theo thứ tự hoành độ x ( nếu hoành độ x bằng nhau thì ưu tiên tung độ y )  B2: Chia tập hợp điểm thành 2 nửa, được phân cách bởi vector trong đó  là điểm đầu tiên trong tập hợp đã sắp xếp và là điểm cuối cùng trong tập hợp đã sắp xếp.  B3: Xác định bao trên   * Bổ sung 2 điểm đầu tiên * Bổ sung điểm thứ 3 ( nếu ba điểm cuối cùng không tạo thành rẽ phải thì xóa điểm giữa)   B4: Xác định bao trên   * + Bổ sung 2 điểm đầu tiên   + Bổ sung điểm thứ 3, nếu ba điểm không tạo thành rẽ phải thì xóa điểm giữa   B5: Xóa điểm đầu và điểm cuối trong  B6: Tập hợp điểm thu được theo kim đồng hồ  \* Để xác định điểm có rẽ phải không thì xác định theo cách sau:   * Cho 3 điểm p, q và r trong không gian 2D:        * Nếu D > 0, thì điểm r nằm trái [p,q] * Nếu D < 0, thì điểm r nằm phải [p, q ] |

1. Viết chương trình xác định bao lồi kèm giải thích chi tiết

|  |
| --- |
| **# Trả lời**: viết câu trả lời vào bên dưới:  #include <iostream>  #include <vector>  #define N 500  using namespace std;  // Định nghĩa cấu trúc điểm 2D  typedef struct  {  double x;  double y;  } point;  // Hàm tính định thức của ba điểm p, q, r  double determinant(point& p, point& q, point& r)  {  double d1, d2;  d1 = q.x \* r.y + p.x \* q.y + p.y \* r.x;  d2 = q.x \* p.y + p.x \* r.y + r.x \* q.y;  return d1 - d2;  }  // Hàm tính Convex Hull  void convexhull(vector<point> a, const int& n)  {  int i, j, j1, j2, n2;  point tg, Lupp[N], Llow[N];  // In ra tập hợp n điểm trong mặt phẳng  cout << "\nTập hợp n điểm trong mặt phẳng: ";  for (i = 1; i <= n; i++)  cout << "(" << a[i].x << "," << a[i].y << "), ";  cout << endl;  // Sắp xếp các điểm theo thứ tự tăng dần của x, sau đó y nếu x bằng nhau  for (i = 1; i <= n - 1; i++)  for (j = n - 1; j >= i; j--)  if ((a[j].x > a[j + 1].x) || ((a[j].x == a[j + 1].x) && (a[j].y > a[j + 1].y)))  {  tg = a[j];  a[j] = a[j + 1];  a[j + 1] = tg;  }  // Tính Upper Hull  Lupp[1] = a[1];  Lupp[2] = a[2];  j1 = 2;  for (i = 3; i <= n; i++)  {  j1++;  Lupp[j1] = a[i];  while ((j1 > 2) && (determinant(Lupp[j1 - 2], Lupp[j1 - 1], Lupp[j1]) > -0.0001))  {  Lupp[j1 - 1] = Lupp[j1];  j1--;  }  }  // Tính Lower Hull  Llow[1] = a[n];  Llow[2] = a[n - 1];  j2 = 2;  for (i = n - 2; i >= 1; i--)  {  j2++;  Llow[j2] = a[i];  while ((j2 > 2) && (determinant(Lupp[j2 - 2], Lupp[j2 - 1], Lupp[j2]) > -0.0001))  {  Lupp[j2 - 1] = Lupp[j2];  j2--;  }  }  // Kết hợp Upper Hull và Lower Hull để có Convex Hull hoàn chỉnh  for (i = 1; i <= j2 - 2; i++)  Lupp[j1 + i] = Llow[i + 1];  n2 = j1 + j2 - 2;  // In ra các đỉnh của Convex Hull  cout << "\nCác đỉnh của Convex Hull: ";  for (i = 1; i <= n2; i++)  cout << "(" << Lupp[i].x << "," << Lupp[i].y << "), ";  cout << endl;  }  int main()  {  int n;  cout << "Nhap so diem:";  cin >> n;  double x, y;  vector<point> points;  point p;  // Nhập các điểm từ người dùng  for (int i = 0; i < n; i++)  {  cin >> x;  cin >> y;  p.x = x;  p.y = y;  points.push\_back(p);  }  // Gọi hàm convexhull để tính Convex Hull  convexhull(points, n);  system("pause");  return 0;  }  **# Trả lời:** Dán kết quả minh họa với 7 điểm cho ở trên |

***Câu 4*** (*3 điểm*): Cho ma trận A

1. Trình bày thuật toán để phân rã ma trận A theo SVD

|  |
| --- |
| **# Trả lời:** Viết điều kiện của ma trận A vào bên dưới :  Để có thể phân rã ma trận A theo SVD thì ma trận A phải thỏa mãn một số điều kiện sau:   * A là ma trận vuông   Thuật toán phân rã A theo SVD gồm các bước sau:   * Tìm ma trận chuyển vị của A * Tìm trị riêng, vector riêng của * Từ các vector riêng tìm được ta xây dựng được ma trận V trong đó cột i của ma trận V là vector riêng thư i. * Xây dựng ma trận D có kích thước bằng ma trận A với  với eigenvalues là ma trận trị riêng và i < length(eigenvalues) * Xây dựng ma trận U * U là ma trận vuông mà số hàng bằng với số hàng của A * Vì A =  => AV = V = UD |

1. Viết hàm thực hiện phân rã SVD và có giải thích chi tiết

|  |
| --- |
| **# Trả lời:** Dán code vào bên dưới:  **# Trả lời:** Thực thi chương trình và dán kết quả cho ví dụ |

Đà Nẵng, ngày 08 tháng 10 năm 2023

|  |  |
| --- | --- |
| **GIẢNG VIÊN BIÊN SOẠN ĐỀ THI** | **TRƯỞNG BỘ MÔN** |
|  | (đã duyệt) |