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  -Kiểm tra điều kiện: Đầu tiên, hãy kiểm tra xem A và B có chia hết cho C hay không. Nếu không, tức là không có nghiệm nguyên và trả về kết quả "vô nghiệm".  -Tìm ước số chung lớn nhất (GCD )của A và B bằng thuật toán Euclid. Gọi GCD(A, B) = D.  -Kiểm tra điều kiện: Nếu C không chia hết cho D, tức là không có nghiệm nguyên, và bạn trả về kết quả "vô nghiệm".  -Tìm một nghiệm nguyên (x₀, y₀) của phương trình sau khi đã chia A và B cho D. Điều này có thể thực hiện bằng cách sử dụng thuật toán Euclid mở rộng để tìm x₀ và y₀ thỏa mãn Ax₀ + By₀ = D.  -Nhân cả phương trình cho C/D để tìm nghiệm cuối cùng (x, y) bằng cách nhân x₀ và y₀ với C/D. |

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  #include <iostream>  using namespace std;  int gcd(int a, int b) {  if (b == 0)  return a;  return gcd(b, a % b);  }  bool solve(int A, int B, int C, int &x, int &y) {  if (C % gcd(A, B) != 0) {  return false;  }  int D = gcd(A, B);  if (C % D != 0) {  return false;  }  int x0, y0;  int a = A / D;  int b = B / D;  int c = C / D;  for (int k = 0; k < D; k++) {  if ((c - a \* k) % b == 0) {  x0 = k;  y0 = (c - a \* k) / b;  break;  }  }  if (x0 < 0 || y0 < 0) {  return false;  }  x = x0;  y = y0;  return true;  }  int main() {  int A, B, C;  cout << "Nhập A, B và C: ";  cin >> A >> B >> C;  int x, y;  if (solve(A, B, C, x, y)) {  cout << "Nghiệm nguyên của phương trình " << A << "x + " << B << "y = " << C << " là (" << x << ", " << y << ")\n";  } else {  cout << "Phương trình vô nghiệm.\n";  }  return 0;  }  **# Trả lời:** Dán kết quả nghiệm của phương trình 5x + 12y = 45. |
|  |

***Câu 2*** (3 *điểm*): Cho ma trận A

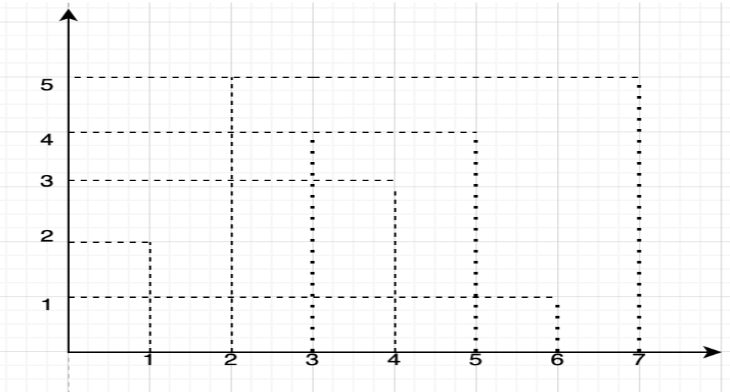
1. Trình bày thuật toán để phân rã ma trận A theo Cholesky

|  |
| --- |
| **# Trả lời:**  1. Khởi tạo ma trận L với các phần tử bằng 0.  2. Duyệt qua các hàng của A từ trên xuống dưới.  3. Đối với mỗi hàng i, tính toán các phần tử của L(i, j) cho j = 1, 2, ..., i.  4. Cập nhật ma trận A bằng cách trừ đi tích của L(i, :) và L(:, i)\*.  5. Lặp lại các bước 2-4 cho đến khi tất cả các hàng của A đã được xử lý. |

1. Viết chương trình để thực hiện phân rã Cholesky

|  |
| --- |
| **# Trả lời:** Dán code vào bên dưới:  import numpy as np  def cholesky\_decomposition(a):  a = np.array(a, float)  n, \_ = np.shape(a)  L = np.zeros\_like(a)  for j in range(n):  for i in range(j, n):  if i == j:  sumk = 0  for k in range(j):  sumk += L[i, k]\*\*2  L[i, j] = np.sqrt(a[i, j] - sumk)  else:  sumk = 0  for k in range(j):  sumk += L[i, k] \* L[j, k]  L[i, j] = (a[i, j] - sumk) / L[j, j]  return L  if \_\_name\_\_ == '\_\_main\_\_':  a = np.array([[7.3, 1, 0], [1, 20, 3.5], [0, 3.5, 2]])  print('----- Matrix A: -----\n' + str(a) + '\n')  v = np.linalg.eigvals(a)  print('----- Matrix V: -----\n' + str(v) + '\n')  L = cholesky\_decomposition(a)  print('----- Matrix L: -----\n' + str(L) + '\n')  **# 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:  1. Tìm điểm có tọa độ x nhỏ nhất và điểm có tọa độ x lớn nhất.  2. Vẽ đường thẳng đi qua hai điểm này.  3. Tìm tất cả các điểm nằm trên hoặc dưới đường thẳng này.  4. Các điểm nằm trên đường thẳng là các điểm thuộc bao lồi.  5. Các điểm nằm dưới đường thẳng không thuộc bao lồi.  Để đảm bảo tính chính xác và hiệu quả của thuật toán, ta có thể sử dụng các kỹ thuật sau:  \* Sử dụng thuật toán tìm kiếm nhị phân để tìm điểm có tọa độ x nhỏ nhất và điểm có tọa độ x lớn nhất.  \* Sử dụng thuật toán sắp xếp để sắp xếp các điểm theo thứ tự tọa độ x.  \* Sử dụng thuật toán tìm giao điểm của hai đường thẳng để tìm giao điểm của đường thẳng đi qua hai điểm có tọa độ x nhỏ nhất và lớn nhất với các đường thẳng đi qua các điểm khác.  \* Sử dụng thuật toán kiểm tra điểm nằm trên hay dưới đường thẳng để kiểm tra xem các điểm khác có nằm trên hay dưới đường thẳng đi qua hai điểm có tọa độ x nhỏ nhất và lớn nhất. |

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

|  |
| --- |
| **# Trả lời**: viết câu trả lời vào bên dưới:  import math  # Định nghĩa một hàm để tính góc giữa hai điểm với điểm gốc là pivot  def angle(p1, p2, pivot):  dx1, dy1 = p1[0] - pivot[0], p1[1] - pivot[1]  dx2, dy2 = p2[0] - pivot[0], p2[1] - pivot[1]  return math.atan2(dy1, dx1) - math.atan2(dy2, dx2)  # Hàm để tìm điểm gốc (pivot) có y-coordinate thấp nhất  def find\_pivot(points):  min\_y = float("inf")  pivot = None  for point in points:  if point[1] < min\_y:  min\_y = point[1]  pivot = point  return pivot  # Hàm thực hiện thuật toán Graham's Scan  def graham\_scan(points):  pivot = find\_pivot(points)  sorted\_points = sorted(points, key=lambda x: (angle(x, pivot, pivot), -x[1], x[0]))    convex\_hull = [pivot, sorted\_points[0]]    for i in range(1, len(sorted\_points)):  while (  len(convex\_hull) > 1 and  angle(sorted\_points[i], convex\_hull[-1], convex\_hull[-2]) <= 0  ):  convex\_hull.pop()  convex\_hull.append(sorted\_points[i])    return convex\_hull  # Danh sách các điểm đầu vào, mỗi điểm là một cặp (x, y)  points = [(0, 3), (1, 1), (2, 2), (4, 4), (0, 0), (1, 2), (3, 1)]  # Gọi hàm Graham's Scan để tìm bao lồi  convex\_hull = graham\_scan(points)  # In kết quả  print("Bao lồi:", convex\_hull)  **# 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 :  Bắt đầu với ma trận A kích thước m x n (m hàng và n cột).  Tính ma trận A^T \* A (chuyển vị của A nhân với A). Đây là một ma trận vuông kích thước n x n.  Tính các giá trị riêng (eigenvalues) và vector riêng (eigenvectors) của ma trận A^T \* A. Điều này có thể được thực hiện bằng cách sử dụng các phương pháp như Phân rã Eigenvalue hoặc SVD ẩn (Implicit SVD).  Sắp xếp các giá trị riêng theo thứ tự giảm dần và chọn các eigenvectors tương ứng để tạo thành ma trận U. Ma trận U là một ma trận trực giao kích thước m x m.  Tính ma trận A \* A^T. Đây là một ma trận vuông kích thước m x m.  Tính các giá trị riêng và eigenvectors của ma trận A \* A^T.  Sắp xếp các giá trị riêng theo thứ tự giảm dần và chọn eigenvectors tương ứng để tạo thành ma trận V. Ma trận V là một ma trận trực giao kích thước n x n.  Tính ma trận Σ bằng cách lấy căn bậc hai của các giá trị riêng của A^T \* A (hoặc A \* A^T) và sắp xếp chúng theo thứ tự giảm dần trên đường chéo của Σ. Ma trận Σ có kích thước m x n, với các giá trị không âm trên đường chéo chính.  Khi hoàn thành, bạn đã phân rã ma trận A thành ba ma trận U, Σ và V^T  -Điều kiện ma trận A : |

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) |