TRƯỜNG ĐẠI HỌC BÁCH KHOA

**KHOA: CÔNG NGHỆ THÔNG TIN**

BỘ MÔN: CÔNG NGHỆ PHẦN MỀM

**ĐỀ THI CUỐI KỲ 1 năm học 2024-2025**

Tên học phần: Toán ứng dụng CNTT

Mã đề: Đ0001 Số tín chỉ: **03**

Phương pháp đánh giá (\*): Tự luận có giám sátThời gian làm bài: **90** phút

☐ Sinh viên được sử dụng tài liệu khi làm bài.

**Họ tên:** Trịnh Minh Đạt…………**Lớp**:…23T\_DT1………………**MSSV**102230180……...

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):

1. Hãy nhập n và tính số nguyên tố gần nhất với f(n) trong dãy Fibonacci

|  |
| --- |
| **# Trả lời:** Mã nguồn hàm tính f(n)  import math  def fibonacci(n):  if n <= 1:  return n  a, b = 0, 1  for \_ in range(2, n + 1):  a, b = b, a + b  return b  **# Trả lời:** Mã nguồn hàm kiểm tra số nguyên tố  def is\_prime(num):  if num <= 1:  return False  for i in range(2, int(math.sqrt(num)) + 1):  if num % i == 0:  return False  return True  **# Trả lời:** Mã nguồn hàm tìm số nguyên tố gần nhất  def nearest\_prime(fib\_num):  lower = fib\_num  while lower > 1 and not is\_prime(lower):  lower -= 1  upper = fib\_num  while not is\_prime(upper):  upper += 1  return lower if (fib\_num - lower) <= (upper - fib\_num) else upper  n = int(input("Nhập n: "))  fib\_n = fibonacci(n)  print(f"Số nguyên tố gần nhất với f({n}) là: {nearest\_prime(fib\_n)}") |

1. Hãy nhập n, m và viết hàm f(n)^n mod m

|  |
| --- |
| **# Trả lời:** Mã nguồn hàm (f(n))^n mod m  def fibonacci(n):  if n <= 1:  return n  a, b = 0, 1  for \_ in range(2, n + 1):  a, b = b, a + b  return b  def power\_mod(base, exp, mod):  result = 1  base %= mod  while exp > 0:  if exp % 2 == 1:  result = (result \* base) % mod  exp //= 2  base = (base \* base) % mod  return result  n = int(input("Nhập n: "))  m = int(input("Nhập m: "))  fib\_n = fibonacci(n)  print(f"(f({n}) ^ n) mod {m} = {power\_mod(fib\_n, n, m)}")  **# Trả lời:** Dán kết quả nghiệm với n = 100, m = 11. |

***Câu 2*** (3 *điểm*): Cho ma trận A

1. Trình bày thuật toán bằng sơ đồ khối để phân rã ma trận A theo SVD

|  |
| --- |
| **# Trả lời:**  Dán sơ đồ khối vào bên dưới |

1. Viết hàm để thực hiện phân rã SVD

|  |
| --- |
| **# Trả lời:** Mã nguồn hàm phân rã SVD vào bên dưới:  import numpy as np  def transpose(matrix):  return np.array(matrix).T  def compute\_at\_a(A):  return np.dot(transpose(A), A)  def compute\_eigenvalues\_and\_vectors(A):  eig\_values, eig\_vectors = np.linalg.eig(A)  return eig\_values, eig\_vectors  def sort\_eigenpairs(eig\_values, eig\_vectors):  idx = np.argsort(eig\_values)[::-1] # Sắp xếp giảm dần  sorted\_eig\_values = eig\_values[idx]  sorted\_eig\_vectors = eig\_vectors[:, idx]  return sorted\_eig\_values, sorted\_eig\_vectors  def compute\_U(A, V, sorted\_eig\_values):  U = np.zeros(A.shape)  for i in range(len(sorted\_eig\_values)):  if sorted\_eig\_values[i] > 1e-10: # Tránh chia cho 0  U[:, i] = np.dot(A, V[:, i]) / np.sqrt(sorted\_eig\_values[i])  return U  def svd(A):  AtA = compute\_at\_a(A)  eig\_values, eig\_vectors = compute\_eigenvalues\_and\_vectors(AtA)  sorted\_eig\_values, V = sort\_eigenpairs(eig\_values, eig\_vectors)  U = compute\_U(A, V, sorted\_eig\_values)  Sigma = np.zeros(A.shape)  for i in range(len(sorted\_eig\_values)):  Sigma[i][i] = np.sqrt(sorted\_eig\_values[i])  return U, Sigma, V  A = np.array([[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]])  U, Sigma, V = svd(A)  print("Ma trận U:")  print(U)  print("\nMa trận Sigma:")  print(Sigma)  print("\nMa trận V:")  print(V)  **# Trả lời:** Dán kết quả khi phân rã SVD của ma trận 7x7 sau |

***Câu 3*** (2 *điểm*): Cho 20 điểm trong không gian 10 chiều với dữ liệu có trong tệp [B.scv](https://drive.google.com/file/d/1KFveXG4ceg68zV_6ecv60ioFYZHVQ-Za/view?usp=sharing), xây dựng thuật toán tính độ tương đồng lớn nhất của 2 điểm trong 20 điểm đã cho bằng độ đo Cosine

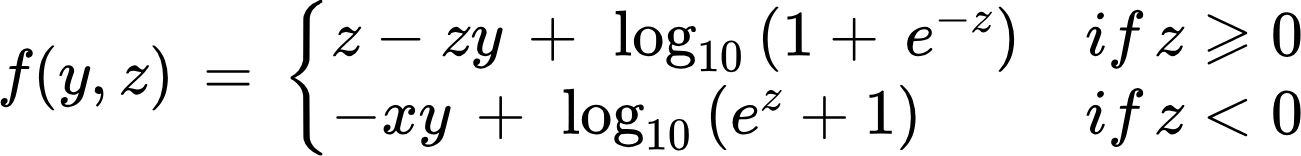
1. *Viết hàm tính độ đo Cosine*

|  |
| --- |
| **# Trả lời:** công thức tính độ đo  **# Trả lời:** Mã nguồn hàm tính độ đo |

1. Xây dựng thuật toán tính độ đo tương đồng lớn nhất của hai điểm

|  |
| --- |
| **# Trả lời**: Trình bày thuật toán đảm bảo thuật toán với độ phức tạp (n log n) với n là số điểm  **# Trả lời:**Mã nguồn hàm thuật toán  **# Trả lời:** Dán kết quả thực nghiệm ( cho biết độ đo và thông tin 2 điểm cụ thể) |

***Câu 4*** (2 *điểm*): Cho hàm



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

|  |
| --- |
| **# Trả lời:** Mã giả  **# Trả lời:** Giải thích khả năng hội tụ Gradient Descent with Momentum |

1. 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:  **# Trả lời**: Dán kết quả chạy chương trình với điểm khởi tạo z = 10, y = 1 vận tốc ban đầu v = 0, tham số học lamda = 0.0001 và tham số vận tốc beta = 0.4. |

***Câu 5*** (1 *điểm*): Cho lịch sử ăn uống các buổi sáng theo trật tự như data = ***['Banh my', 'Pho', 'Bun', 'Pizza', 'Banh my', 'Pho', 'Bun', 'Pizza', 'Pizza', 'Banh my', 'Banh my', 'Pho', 'Pho', 'Pho', 'Bun', 'Pho']***  *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:

1. Viết hàm xây dựng ma trận trạng thái P

|  |
| --- |
| **# Trả lời:** Dán code  **# Trả lời:** Kết quả ma trận trạng thái P |

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

|  |
| --- |
| **# Trả lời**: Dán code  **# Trả lời**: Kết quả xác suất: |

Đà Nẵng, ngày 4 tháng 12 năm 2024

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