**TÓM TẮT NỘI DUNG MÔN HỌC TOÁN ỨNG DỤNG**

[**CHƯƠNG 1: NUMBER THEORY** 2](#_Toc121901976)

[**1.** **Số nguyên tố và hợp số** 2](#_Toc121901977)

[**2.** **Phân tích thành thừa số nguyên tố** 2](#_Toc121901978)

[**3.** **Số các ước, tổng và tích của chúng** 2](#_Toc121901979)

[**4.** **Số hoàn hảo** 3](#_Toc121901980)

[**5.** **Thuật toán sàng nguyên tố Eratosthenes** 4](#_Toc121901981)

[**6.** **Hàm tìm ước chung lớn nhất, bội chung nhỏ nhất** 6](#_Toc121901982)

[**7.** **Bài toán giải phương trình Dioplantine** 7](#_Toc121901983)

[**8.** **Định lý thặng dư trung hoa** 9](#_Toc121901984)

[**CHƯƠNG 2: LINEAR ALGEBRA** 12](#_Toc121901985)

[**1.** **Ma trận nghịch đảo** 12](#_Toc121901986)

[**2.** **Tính định thức** 14](#_Toc121901987)

[**3.** **Phân rã ma trận Chelosky** 15](#_Toc121901988)

[**4.** **Phân rã ma trận theo SVD** 18](#_Toc121901989)

[**5.** **Phân rã ma trận theo vecto riêng và trị riêng** 22](#_Toc121901990)

[**CHƯƠNG 3: GEOMETRY** 33](#_Toc121901991)

[**1.** **Bài toán bao lồi** 33](#_Toc121901992)

[**2.** **Bài toán cặp điểm gần nhất** 35](#_Toc121901993)

[**CHƯƠNG 4: OPTIMIZATIONS** 39](#_Toc121901994)

[**1.** **Bài toán 1: Gradient descent cho hàm 1 biến** 39](#_Toc121901995)

[**2.** **Bài toán 2: Gradient descent cho hàm nhiều biến** 40](#_Toc121901996)

[**3.** **Bài toán 3: Gradient descent momentum cho hàm 1 biến** 42](#_Toc121901997)

[**4.** **Bài toán 4: Gradient descent with momentum với hàm nhiều biến** 46](#_Toc121901998)

[**5.** **Phương pháp đơn hình** 46](#_Toc121901999)

[**6.** **Phương pháp đơn hình 2 pha** 54](#_Toc121902000)

[**CHƯƠNG 5: PROBABILITY** 62](#_Toc121902001)

[**Chuỗi markov** 62](#_Toc121902002)

GIÁO VIÊN: NGUYỄN VĂN HIỆU

SINH VIÊN: NGUYỄN PHƯỚC ĐẠI TOÀN

MSSV: 102200035. Lớp: 20T1

# **CHƯƠNG 1: NUMBER THEORY**

1. **Số nguyên tố và hợp số**

bool checkIsPrime (int n) {

if (n < 2) return false;

for (int i = 2; i <= sqrt(n); i++) {

if (n % i == 0)

return false;

}

return true;

}

1. **Phân tích thành thừa số nguyên tố**

void factorize (int n) {

int temp = 0;

int i = 2;

while (n > 1) {

if(checkIsPrime(i) && n % i == 0) {

while (n % i == 0) {

temp++;

n /= i;

}

if (n > 1) {

cout<<i<<"^"<<temp<<" . ";

} else {

cout<<i<<"^"<<temp;

}

temp = 0;

}

i++;

}

cout<<endl;

}

1. **Số các ước, tổng và tích của chúng**
2. **Số các ước số**

int SoCacUocSo (int n) {

    int temp = 0;   // bien tam de luu so mu

    int i = 2;      // bien chay phan tich thanh thua so nguyen to

    int result = 1; // so cac uoc so

    while (n > 1) {

        if(checkIsPrime(i) && n % i == 0) {

            while (n % i == 0) {

                temp++;

                n /= i;

            }

            result \*= (temp + 1);

            temp = 0;

        }

        i++;

    }

    return result;

}

1. **Tổng các ước số**

**void** TongCacUocSo (**int** n) {

cout<<"Tong cac uoc so cua "<<n<<": ";

**int** temp = 0; // bien tam de luu so mu

**int** i = 2; // bien chay phan tich thanh thua so nguyen to

**int** result = 1; // tong cac uoc so

**while** (n > 1) {

if(checkIsPrime(i) && n % i == 0) {

**while** (n % i == 0) {

temp++;

n /= i;

}

result \*= (pow (i, temp + 1) - 1) / (i - 1);

temp = 0;

}

i++;

}

cout<<result<<endl;

}

1. **Tích các ước số**

**void** TichCacUocSo (**int** n) {

cout<<"Tich cac uoc so cua "<<n<<": ";

cout<<pow (n, SoCacUocSo(n) \* 1.0 / 2) <<endl;

}

1. **Số hoàn hảo**

**bool** checkIsPerfectNumber (**int** n) {

**if** (n < 6)

**return** false;

**int** sum = 1;

**for**(**int** i = 2; i <= sqrt(n); i++) {

if(n % i == 0) {

sum += i;

sum += n / i;

}

}

return sum == n;

}

1. **Thuật toán sàng nguyên tố Eratosthenes**

Viết chương trình nhập từ bàn phím N = 210000000, thực hiện các công việc sau:

1. *(1 điểm)* Trình bày thuật toán ràng Eratosthenes để tìm các số nguyên tố

|  |
| --- |
| **# Trả lời:** Trình bày thuật toán ở bên dưới:  **Mô tả:** Ý tưởng: ta loại bỏ các bội số của 2, 3, 4, … đến N để loại bỏ các bội số, sau đó ta sẽ tìm được các số nguyên tố cần tìm  **Input:** Số nguyên dương lớn hơn 1  **Thuật toán:**  Bước 1: Khởi tạo mảng check có N + 1 phần tử với giá trị ban đầu là true  Bước 2: Lặp i từ 2 đến N với bước nhảy i++;   * Lặp j từ 2 \* i đến N với bước nhảy j += i để loại bỏ các bội số của i (hợp số) * Xét check[j] = false;   Bước 3: In ra các số nguyên số i nhỏ hơn N với check[i] = true  **Ouput:** Tìm các số nguyên tố nhỏ hơn N |

1. *(1 điểm)* viết chương trình mô phỏng phương pháp Eratosthenes để tìm các số nguyên tố không vuợt quá N.

|  |
| --- |
| **# Trả lời:** Dán code vào bên dưới  #include<iostream>  using namespace std;  // Thuat toan sang nguyen to Eratosthenes  void eratosthenes(int n) {  if (n < 2) return;  bool \*check = new bool[n + 1];  for (int i = 2; i <= n; i++)  check[i] = true;    // Loc tat ca cac so la boi cua i  for (int i = 2; i <= n; i++) {  if(check[i]) {  for (int j = 2 \* i; j <= n; j += i) {  check[j] = false;  }  }  }    // In ra cac so nguyen to  for(int i = 2; i <= n; i++) {  if(check[i]) {  cout<<i<<" ";  }  }  }  int main () {  eratosthenes (210000000);  return 0;  }  **# Trả lời:** Dán kết quả thực thi vào bên dưới: |

1. **Hàm tìm ước chung lớn nhất, bội chung nhỏ nhất**
2. **Ước chung lớn nhất:**

**int** GCD (**int** a, **int** b) {

**while** (a \* b != 0) {

if (a < b) b = b % a;

else a = a % b;

}

return a + b;

}

Hoặc hàm Euclid:

**int** GCD\_Euclid (**int** a, **int** b) {

if (b == 0)

return a;

return GCD\_Euclid (b, a % b);

}

1. **Bội chung nhỏ nhất**

**int** LCM (**int** a, **int** b) {

return a \* b / GCD (a, b);

}

1. **Bài toán giải phương trình Dioplantine**

|  |
| --- |
| **Mô tả:**  Cho phương trình Diophantine:  với  Tìm cặp nghiệm nguyên x, y thoả mãn phương trình trên.  **Input:** Các hệ số của phương trình  **Thuật toán:**  Bước 1: Tìm  Bước 2: Kiểm tra phương trình có nghiệm hay vô nghiệm   * Nếu  không chia hết cho  thì phương trình vô nghiệm và dừng thuật toán. * Ngược lại thì   Bước 3: Tìm hệ số  sao cho  Bước 4: Kết luận   * Nghiệm của phương trình là: * Nghiệm tổng quát:   **Output:** Nghiệm nguyên  và |
| **Code:**  #include<iostream>  using namespace std;  typedef pair<int, int> vector;  #define k first  #define l second  int gcd(int a, int b) {  while(a \* b != 0) {  if(a < b) b = b % a;  else a = a % b;  }  return a + b;  }  vector extended\_gcd(int a, int b) {  vector qr, st;  if (b == 0)  return vector(1, 0);  else {  qr = vector(a / b, a % b);  st = extended\_gcd(b, qr.l);  return vector(st.l, st.k - qr.k \* st.l);  }  }  void solveDiopantine(int a, int b, int c) {  int d = gcd(a, b);  if (c % d != 0) {  printf("Phuong trinh %d.x + %d.y = %d khong co nghiem nguyen\n", a, b, c);  return;  }    int q = c / d;  vector kl = extended\_gcd(a, b);    // Nghiem rieng cua phuong trinh  int x = q \* kl.k;  int y = q \* kl.l;  printf("Nghiem cua phuong trinh %d.x + %d.y = %d la:\n", a, b, c);  printf("Nghiem rieng: x = %d, y = %d \n", x, y);  printf("Nghiem tong quat: x = %d + %d.r, y = %d - %d.r", x, b/d, y, a/d);  }  int main() {  solveDiopantine(3, 6, 22);  solveDiopantine(7, 11, 13);  return 0;  } |
| **Kết quả:** |

1. **Định lý thặng dư trung hoa**

|  |
| --- |
| **Mô tả:**  Với phương trình  ,  Hệ phương trình có nghiệm duy nhất với  Trong đó:        **Input:** Các hệ số  với  **Thuật toán:**  Bước 1: Khởi tạo  Bước 2: Lặp  đến  • Tính  • Nếu  thì continue đến vòng lặp tiếp theo  • Lặp tìm  đến   * Nếu thì  và thoát khỏi vòng lặp j * x = x % M;   Bước 4: Trả về  (nghiệm của hệ phương trình)  **Output:** Nghiệm của hệ phương trình: |
| **Code:**  #include<iostream>  using namespace std;  int GiaiThangDu(int a[][2], int n) {  int M = 1, Mi;  int x = 0;    for(int i = 0; i < n; i++) {  // M = m1 \* m2 \* ...  M \*= a[i][1];  }  int z = 1;  for(int i = 0; i < n; i++) {  // Tim M1, M2, M3, ....  Mi = M / a[i][1];  z = Mi % a[i][1];  if(z == 0) continue;  // Tim y1, y2, y3,...  for(int yi = 1; yi < Mi; yi++) {  if(z \* yi % a[i][1] == 1) {  x += a[i][0] \* Mi \* yi;  break;  }  }  x %= M;  }  return x;  }  int main() {  int a[][2] = {  {7, 9},  {5, 7},  {2, 5}  };  for(int i = 0; i < 3; i++) {  printf("a%d = %d\t", i, a[i][0]);  printf("m%d = %d", i, a[i][1]);  cout<<endl;  }    cout<<"x = "<<GiaiThangDu(a, sizeof(a) / sizeof(a[0]));    return 0;  } |
| **Kết quả:** |

**CHƯƠNG 2: LINEAR ALGEBRA**

1. **Ma trận nghịch đảo**

#include<iostream>

using namespace std;

int main() {

int n, check = 1;

cout<<"Nhap cap ma tran: ";

cin>>n;

float a[n][2 \* n];

float b[n][n];

cout<<"Nhap ma tran: "<<endl;

for(int i = 0; i < n; i++) {

for(int j = 0; j < n; j++) {

cout<<"a["<<i<<"]["<<j<<"] = ";

cin>>a[i][j];

}

}

cout<<"Ma tran vua nhap: "<<endl;

for(int i = 0; i < n; i++) {

for(int j = 0; j < n; j++) {

cout<<a[i][j]<<"\t";

}

cout<<endl;

}

// Gop ma tran don vi vao A

for (int i = 0; i < n; i++) {

for (int j = n; j < 2 \* n; j++) {

if (j - i == n)

a[i][j] = 1;

else

a[i][j] = 0;

}

}

int t = 0;

while (t < n)

{

int k;

if (a[t][t] == 0)

{

check = 0;

}

else if (a[t][t] != 1)

{

k = a[t][t];

for(int i=t;i<2\*n ;i++)

{

a[t][i] /= k;

}

}

for (int i = t + 1; i < n; i++)

{

k = a[i][t];

for (int j = t; j < 2 \* n; j++)

a[i][j] = a[i][j] - k \* a[t][j];

}

t++;

}

t--;

while (t > 0)

{

for (int i = t - 1; i >= 0; i--)

{

int k = a[i][t];

for (int j = t; j < 2 \* n; j++)

a[i][j] = a[i][j] - k \* a[t][j];

}

t--;

}

if (check == 1)

{

cout<<"Ma tran nghich dao la: "<<endl;

for(int i=0;i<n;i++)

{

for (int j = n; j < n \* 2; j++)

cout << a[i][j] << " ";

cout << "\n";

}

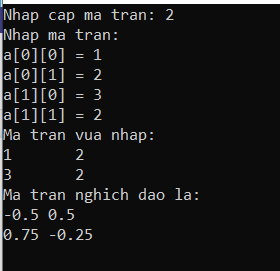
}

else

cout << "Khong ton tai ma tran nghich dao !!!";

return 0;

}



1. **Tính định thức**

float TinhDinhThuc(float a[][100], int n) {

float b[n][100];

int i, j, k, temp;

for(i = 0; i < n; i++) {

for(int j = 0; j < n; j++) {

b[i][j] = a[i][j];

}

}

// Lap n - 1 lan de chuyen ve ma tran tam giac

float det = 1;

for(i = 0; i < n - 1; i++) {

// Neu gap phan tu tren duong cheo bang 0 thi doi cho hang khac

if (b[i][i] == 0) {

k = i + 1;

while (k < n && b[k][i] == 0)

k++;

// Neu khong tim thay hang doi

if (k == n) {

return 0;

}

// Doi hang i voi hang k -> lap cot j tu i den n

for (j = i; j < n; j++) {

temp = b[i][j];

b[i][j] = b[k][j];

b[k][j] = temp;

}

}

float t = b[i][i];

// Lap tu hang i + 1 den n - 1

for(j = i + 1; j < n; j++) {

// Lap cot 0 den n - 1

float g = b[j][i];

for(int k = 0; k < n; k++) {

b[j][k] -= g / t \* b[i][k];

}

}

det \*= b[i][i];

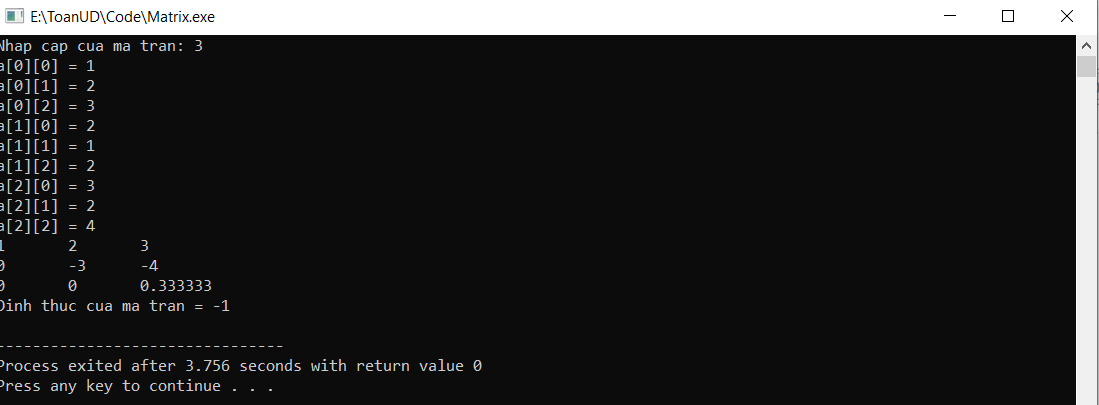
}

det \*= b[n-1][n-1];

Xuat(b, n);

return det;

}



1. **Phân rã ma trận Chelosky**

|  |
| --- |
| **Mô tả:**  Cho ma trận A đầu vào để phân rã theo Chelosky     * Điều kiện của ma trận đầu vào A:   + Ma trận vuông  + Ma trận đối xứng  + Ma trận xác định dương   * Khi đó ma trận A phân rã theo công thức:      * Kết quả:     Hay  **Input:** Các hệ số của ma trận A  **Thuật toán:**    **Output:** Ma trận phân rã: |
| **Code:**  #include<iostream>  #include<math.h>  #include<cstring>  using namespace std;  const int MAX = 100;  void Cholesky\_Decomposition(int matrix[][MAX],int n)  {  int L[n][n];  memset(L, 0, sizeof(L));  // Code thuat toan Cholesky  for (int i = 0; i < n; i++) {  for (int j = 0; j <= i; j++) {  int sum = 0;    if (i == j) {  for (int k = 0; k < j; k++)  sum += L[j][k] \* L[j][k];  L[j][j] = sqrt(matrix[j][j] - sum);  } else {  for (int k = 0; k < j; k++)  sum += (L[i][k] \* L[j][k]);  L[i][j] = (matrix[i][j] - sum) / L[j][j];  }  }  }  cout << "\t\t" << " Ma tran tam giac duoi"  << "\t\t\t\t\t" << "Ma tran tam giac tren" << endl << endl;  for (int i = 0; i < n; i++) {  for (int j = 0; j < n; j++)  cout <<"\t" << L[i][j] << "\t";  cout << "\t";    for (int j = 0; j < n; j++)  cout <<"\t" << L[j][i] << "\t";  cout << endl;  }  }  int main()  {  int n = 3;  int matrix[][MAX] = { { 4, 12, -16 },  { 12, 37, -43 },  { -16, -43, 98 } };  cout<<"Ma tran khoi tao la: "<<endl;  for(int i = 0; i < 3; i++) {  for(int j = 0; j < 3; j++) {  cout<<matrix[i][j]<<"\t";  }  cout<<endl;  }  cout<<"Phan ra ma tran Cholesky"<<endl;  Cholesky\_Decomposition(matrix, n);  return 0;  } |
| **Kết quả:** |

## **Phân rã ma trận theo SVD**

|  |
| --- |
| **Mô tả:**   * Điều kiện của ma trận A:   + Ma trận thực  + Ma trận không suy biến  **Input:** Ma trận thực không suy biến  **Thuật toán:**      **Output:** Các ma trận |
| **Code:**  #include <bits/stdc++.h>  using namespace std;  #define MAX 100  void transpose(float A[][MAX], float B[][MAX], int n)  {  for (int i = 0; i < n; i++)  {  for (int j = 0; j < n; j++)  B[i][j] = A[j][i];  }  }  void swap(float &a, float &b)  {  float temp;  temp = a;  a = b;  b = temp;  }  void printMatrix(float A[][MAX], int n)  {  for (int i = 0; i < n; i++)  {  for (int j = 0; j < n; j++)  cout << setprecision(10) << A[i][j] << " ";  cout << endl;  }  }  void squareMatrixProduct(float A[][MAX], float B[][MAX], float C[][MAX], int n)  {  for (int i = 0; i < n; i++)  {  for (int j = 0; j < n; j++)  {  C[i][j] = 0;  for (int k = 0; k < n; k++)  {  C[i][j] += A[i][k] \* B[k][j];  }  }  }  }  void SolvedCubic(float C[], float X[])  {  float delta, k;  delta = pow(C[1], 2) - 3 \* C[0] \* C[2];  k = (9 \* C[0] \* C[1] \* C[2] - 2 \* pow(C[1], 3) - 27 \* pow(C[0], 2) \* C[3]) / (2 \* sqrt(pow(fabs(delta), 3)));  if (delta > 0)  {  if (fabs(k) <= 1)  {  X[0] = (2 \* sqrt(delta) \* cos(acos(k) / 3) - C[1]) / (3 \* C[0]);  X[1] = (2 \* sqrt(delta) \* cos(acos(k) / 3 - (2 \* M\_PI / 3)) - C[1]) / (3 \* C[0]);  X[2] = (2 \* sqrt(delta) \* cos(acos(k) / 3 + (2 \* M\_PI / 3)) - C[1]) / (3 \* C[0]);  }  if (fabs(k) > 1)  {  X[0] = ((sqrt(delta) \* fabs(k)) / (3 \* C[0] \* k)) \* (pow((fabs(k) + sqrt(pow(k, 2) - 1)), 1.0 / 3) + pow((fabs(k) - sqrt(pow(k, 2) - 1)), 1.0 / 3)) - (C[1] / (3 \* C[0]));  }  }  else if (delta == 0)  {  X[0] = (-C[1] - pow(-(pow(C[1], 3) - 27 \* C[0] \* C[0] \* C[3]), 1.0 / 3)) / (3 \* C[0]);  }  else  X[0] = (sqrt(fabs(delta)) / (3 \* C[0])) \* (pow((k + sqrt(k \* k + 1)), 1.0 / 3) - pow(-(k - sqrt(k \* k + 1)), 1.0 / 3)) - (C[1] / (3 \* C[0]));  }  void EigenDecomposition(float A[][MAX], float P[][MAX], float D[][MAX], int n)  {  float M[MAX][MAX], M1[MAX][MAX], B[MAX][MAX], C[MAX][MAX];  for (int i = 0; i < n; i++)  {  for (int j = 0; j < n; j++)  {  if (j == i)  C[i][j] = 1;  else  C[i][j] = 0;  }  }  for (int k = n - 2; k >= 0; k--)  {  for (int i = 0; i < n; i++)  {  for (int j = 0; 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 (j == k)  M[i][j] = 1.0 / A[k + 1][k];  else  M[i][j] = -A[k + 1][j] / A[k + 1][k];  }  }  }  squareMatrixProduct(A, M, B, n);  squareMatrixProduct(M1, B, A, n);  squareMatrixProduct(C, M, B, n);  for (int i = 0; i < n; i++)  {  for (int j = 0; j < n; j++)  C[i][j] = B[i][j];  }  }  float heso[MAX] = {1, -A[0][0], -A[0][1], -A[0][2]};  cout << endl;  float X[MAX];  SolvedCubic(heso, X);  for (int i = 0; i < n; i++)  {  for (int j = 0; j < n; j++)  {  if (i == j)  D[i][j] = X[i];  else  D[i][j] = 0;  }  }  float T[MAX][MAX];  for (int i = 0; i < n; i++)  {  for (int j = 0; j < n; j++)  {  T[i][j] = pow(X[j], n - i - 1);  }  }  float P1[MAX][MAX];  squareMatrixProduct(B, T, P, n);  for (int j = 0; j < n; j++)  {  float p = 0;  for (int i = 0; i < n; i++)  p += pow(P[i][j], 2);  for (int i = 0; i < n; i++)  P[i][j] = P[i][j] / sqrt(p);  }  }  void SVD(float A[][MAX], int n)  {  float At[MAX][MAX];  transpose(A, At, n);  float S[MAX][MAX], P[MAX][MAX], D[MAX][MAX], U[MAX][MAX];  squareMatrixProduct(At, A, S, n);  EigenDecomposition(S, P, D, n);  float Vt[MAX][MAX];  transpose(P, Vt, n);  for (int i = 0; i < n; i++)  {  for (int j = 0; j < n; j++)  D[i][j] = sqrt(D[i][j]);  }  squareMatrixProduct(A, P, U, n);  for (int i = 0; i < n; i++)  {  for (int j = 0; j < n; j++)  {  U[j][i] \*= 1.0 / D[i][i];  }  }  // in ma tran  cout << "Singular value decomposition: " << endl;  cout << "U" << endl;  for (int i = 0; i < n; i++)  {  // Ma tran U  for (int j = 0; j < n; j++)  {  if (j == 0)  cout << setprecision(5) << (-1 \* U[i][j]) << "\t\t";  else  cout << setprecision(5) << U[i][j] << "\t\t";  }  cout << endl;  }  cout << endl;  cout << "D" << endl;  for (int i = 0; i < n; i++)  {  // Ma tran S (ma tr?n D)  for (int j = 0; j < n; j++)  cout << setprecision(5) << D[i][j] << "\t\t";  cout << endl;  }  cout << endl;  cout << "Vt" << endl;  for (int i = 0; i < n; i++)  {  // Ma tran Vt  for (int j = 0; j < n; j++)  {  if (i == 0)  cout << setprecision(5) << (-1 \* Vt[i][j]) << "\t\t";  else  cout << setprecision(5) << Vt[i][j] << "\t\t";  }  cout << endl;  }  }  int main()  {  int n = 3;  float A[][MAX] = {{13, 15, 17},  {15, 9, 7},  {17, 7, 5}};  float P[MAX][MAX];  cout << "Ma tran A:" << endl;  printMatrix(A, n);  cout << "-------------------" << endl;  cout << "Phan ra SVD: A = U \* S \* Vt" << endl;  SVD(A, n);  return 0;  }  **Kết quả:** |

1. **Phân rã ma trận theo vecto riêng và trị riêng**

|  |
| --- |
| **Mô tả:**   * Điều kiện của ma trận đầu vào A:   + Ma trận vuông   * Ma trận A có thể phân rã thành các ma trận:   Trong đó:  +  vector riêng  +  : ma trận đường chéo với các chỉ số riêng  **Input:** Ma trận đầu vào A  **Output:** Phân rã thành các ma trận theo vector riêng và chỉ số riêng |
| **Code:**  #include <stdio.h>  #include <iostream>  #include <cmath>  using namespace std;  void nhap(float a[][100], int &n)  {  cout << "Nhap chieu ma tran: ";  cin >> n;  cout << "Nhap ma tran " << endl;  for (int i = 0; i < n; i++)  for (int j = 0; j < n; j++)  {  cin >> a[i][j];  }  }  void xuat(float a[][100], int n)  {  for (int i = 0; i < n; i++)  {  for (int j = 0; j < n; j++)  printf("%5.3f ", a[i][j]);  cout << endl;  }  }  void gan(float a[][100], float b[][100], int n)  {  for (int i = 0; i < n; i++)  {  for (int j = 0; j < n; j++)  a[i][j] = b[i][j];  }  }  void MTDonVi(float a[][100], int n)  {  for (int i = 0; i < n; i++)  {  for (int j = 0; j < n; j++)  a[i][j] = (i == j) ? 1 : 0;  }  }  void nhanMT(int n, int p, int m, float a[][100], float b[][100], float c[][100])  {  for (int i = 0; i < n; i++)  for (int j = 0; j < m; j++)  {  c[i][j] = 0;  for (int k = 0; k < p; k++)  c[i][j] += a[i][k] \* b[k][j];  }  }  void triRieng(float a[][100], float M[][100], int n)  {  float m[100][100], m1[100][100], b[100][100], temp[100][100];  for (int k = n - 2; k >= 0; k--)  {  for (int i = 0; i < n; i++)  {  for (int j = 0; j < n; j++)  {  if (i != k)  {  if (i != j)  m[i][j] = m1[i][j] = 0;  else  m[i][j] = m1[i][j] = 1;  }  else  {  m1[i][j] = a[k + 1][j];  if (j == k)  m[i][j] = 1 / a[k + 1][k];  else  m[i][j] = -a[k + 1][j] / a[k + 1][k];  }  }  }  nhanMT(n, n, n, a, m, b);  nhanMT(n, n, n, m1, b, a);  gan(temp, M, n);  nhanMT(n, n, n, temp, m, M);  }  }  void phuongTrinh(float a[][100], float pt[], int n)  {  pt[0] = 1;  for (int i = 0; i < n; i++)  {  pt[i + 1] = -a[0][i];  }  }  void giaiPTBacBa(float pt[], float nghiem[])  {  const double PI = 4.0 \* atan(1.0);  double p = (pt[1] \* pt[1] - 3.0 \* pt[0] \* pt[2]) / (9.0 \* pt[0] \* pt[0]);  double q = (9.0 \* pt[0] \* pt[1] \* pt[2] - 27.0 \* pt[0] \* pt[0] \* pt[3] - 2.0 \* pt[1] \* pt[1] \* pt[1]) / (54.0 \* pt[0] \* pt[0] \* pt[0]);  double offset = pt[1] / (3.0 \* pt[0]);  double dis = p \* p \* p - q \* q;  if (dis > 0)  {  double theta = acos(q / (p \* sqrt(p)));  double r = 2.0 \* sqrt(p);  for (int i = 0; i < 3; i++)  nghiem[i] = r \* cos((theta + 2.0 \* i \* PI) / 3.0) - offset;  }  else  {  double g1 = cbrt(q + sqrt(-dis));  double g2 = cbrt(q - sqrt(-dis));  nghiem[0] = g1 + g2 - offset;  double re = -0.5 \* (g1 + g2) - offset;  double im = (g1 - g2) \* sqrt(3.0) / 2.0;  if (dis == 0.0)  {  nghiem[1] = re;  }  }  }  void giaiPTBacHai(float pt[], float nghiem[])  {  float delta = pt[1] \* pt[1] - 4 \* pt[0] \* pt[2];  if (delta < 0)  {  nghiem[0] = nghiem[1] = 0.0;  }  else if (delta == 0)  {  nghiem[0] = -pt[1] / (2 \* pt[0]);  nghiem[1] = 0.0;  }  else  {  delta = sqrt(delta);  nghiem[0] = (-pt[1] + delta) / (2 \* pt[0]);  nghiem[1] = (-pt[1] - delta) / (2 \* pt[0]);  }  }  void giaiPT(float pt[], float nghiem[], int n)  {  if (n == 2)  giaiPTBacHai(pt, nghiem);  else if (n == 3)  giaiPTBacBa(pt, nghiem);  }  void vtRieng(float V[][100], float M[][100], float nghiem[], int n)  {  for (int k = 0; k < n; k++)  {  float temp[100][100], kq[100][100];  for (int i = 0; i < n; i++)  {  temp[i][0] = pow(nghiem[k], n - 1 - i);  }  nhanMT(n, n, 1, M, temp, kq);  for (int i = 0; i < n; i++)  {  V[i][k] = kq[i][0];  }  }  }  void maTranDuongCheo(float lamda[][100], float nghiem[], int n)  {  for (int i = 0; i < n; i++)  {  for (int j = 0; j < n; j++)  {  lamda[i][j] = (i == j) ? nghiem[i] : 0;  }  }  }  void maTranBD(float bd[][200], float V1[][100], int n)  {  for (int i = 0; i < n; i++)  {  for (int j = 0; j < 2 \* n; j++)  {  if (j == i + n)  {  bd[i][j] = 1;  }  else if (j < n)  {  bd[i][j] = V1[i][j];  }  else  {  bd[i][j] = 0;  }  }  }  }  void nghichDaoMaTran(float V1[][100], int n)  {  float bd[100][200];  maTranBD(bd, V1, n);  for (int k = 0; k < n; k++)  {  float heSo = bd[k][k];  for (int i = 0; i < 2 \* n; i++)  {  bd[k][i] /= heSo;  }  for (int i = 0; i < n; i++)  {  if (i != k)  {  heSo = bd[i][k];  for (int j = 0; j < 2 \* n; j++)  {  bd[i][j] = bd[i][j] - bd[k][j] \* heSo;  }  }  }  }  for (int i = 0; i < n; i++)  {  for (int j = n; j < 2 \* n; j++)  V1[i][j - n] = bd[i][j];  }  }  void phanRaMaTran(int n)  {  float a[100][100], M[100][100], V[100][100], lamda[100][100], V1[100][100], result[100][100];  float pt[100], nghiem[100];  nhap(a, n);  MTDonVi(M, n);  triRieng(a, M, n);  phuongTrinh(a, pt, n);  giaiPT(pt, nghiem, n);  vtRieng(V, M, nghiem, n);  cout << "Ma tran vecto rieng V: " << endl;  xuat(V, n);  maTranDuongCheo(lamda, nghiem, n);  cout << "Ma tran duong cheo lamda: " << endl;  xuat(lamda, n);  gan(V1, V, n);  nghichDaoMaTran(V1, n);  cout << "Ma tran nghich dao cua V: " << endl;  xuat(V1, n);  float temp[100][100];  nhanMT(n, n, n, V, lamda, temp);  nhanMT(n, n, n, temp, V1, result);  cout << "Ma tran phan ra theo chi so rieng: " << endl;  xuat(result, n);  }  int main()  {  int n;  phanRaMaTran(n);  return 0;  } |
| **Kết quả:** |

**CHƯƠNG 3: GEOMETRY**

1. **Bài toán bao lồi**

|  |
| --- |
| **Mô tả:** Từ tập các điểm theo yêu cầu đề bài ta tìm bao lồi (tập lồi nhỏ nhất chứa tất cả các điểm đề bài  **Input:** Toạ độ các điểm theo đề bài  **Thuật toán:**  Bước 1: Sắp xếp các điểm theo toạ độ x tăng dần (nếu cùng toạ độ x thì ta xếp theo toạ độ y tăng dần)  Xác định bao trên Lupper (bước 2, 3)  Bước 2: Bổ sung P1 và P2 vào bao trên Lupper với P1 là điểm bắt đầu  Bước 3: Bổ sung điểm thứ 3 (nếu 3 điểm cuối cùng không tạo thành rẻ phải thì xoá điểm giữa)  Xác định bao dưới Llower (bước 4, 5, 6)  Bước 4: Bổ sung Pn và Pn-1 vào bao dưới Llower với Pn là điểm bắt đầu  Bước 5: Bổ sung điểm thứ 3, nếu 3 điểm không tạo thành rẻ phải thì điểm giữa  Bước 6: Xoá điểm đầu và điểm cuối trong bao dưới Llower  Bước 7: Tập hợp điểm thu được theo chiều kim đồng hồ: Lupper U Llower  **Output:** Bao lồi (tập lồi nhỏ nhất chứa tất cả các điểm) |
| **Code:**  #include<iostream>  using namespace std;  typedef struct {  double x;  double y;  } Point;  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;  }  void convexhull (Point \*a, const int &n) {  Point t, Lup[n], Llow[n];    // Buble Sort  for (int i = 0; i < n; i++) {  for (int j = n - 2; 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)) {  t = a[j];  a[j] = a[j + 1];  a[j + 1] = t;  }  }  }    // Find Lupper  Lup[0] = a[0];  Lup[1] = a[1];  int j1 = 1;  for (int i = 2; i < n; i++) {  j1++;  Lup[j1] = a[i];  while (j1 >= 2 && determinant (Lup[j1 - 2], Lup[j1 - 1], Lup[j1]) > -0.0001) {  Lup[j1 - 1] = Lup[j1];  j1--;  }  }    // Find Llower  Llow[0] = a[n - 1];  Llow[1] = a[n - 2];  int j2 = 1;  for (int i = n - 3; i >= 0; i--) {  j2++;  Llow[j2] = a[i];  while (j2 >= 2 && determinant (Llow[j2 - 2], Llow[j2 - 1], Llow[j2]) > -0.0001) {  Llow[j2 - 1] = Llow[j2];  j2--;  }  }  // convex hull full  for (int i = 1; i <= j2 - 1; i++) {  Lup[j1 + i] = Llow[i];  }  cout<<"Bao loi: \n";  for (int i = 0; i < j1 + j2; i++) {  printf ("(%lf, %lf) \n", Lup[i].x, Lup[i].y);  }  cout<<endl;  }  int main () {  Point a[] = {  {1, 2},  {2, 5},  {3, 4},  {4, 3},  {5, 4},  {6, 1},  {7, 5}  };  convexhull (a, sizeof(a)/sizeof(a[0]));  return 0;  } |
| Kết quả: |

1. **Bài toán tính diện tích bao lồi**

|  |
| --- |
| **Mô tả:** Từ tập các điểm theo yêu cầu đề bài ta tìm bao lồi (tập lồi nhỏ nhất chứa tất cả các điểm đề bài  **Input:** Toạ độ các điểm theo đề bài  **Thuật toán:**  Bước 1: Sắp xếp các điểm theo toạ độ x tăng dần (nếu cùng toạ độ x thì ta xếp theo toạ độ y tăng dần)  Xác định bao trên Lupper (bước 2, 3)  Bước 2: Bổ sung P1 và P2 vào bao trên Lupper với P1 là điểm bắt đầu  Bước 3: Bổ sung điểm thứ 3 (nếu 3 điểm cuối cùng không tạo thành rẻ phải thì xoá điểm giữa)  Xác định bao dưới Llower (bước 4, 5, 6)  Bước 4: Bổ sung Pn và Pn-1 vào bao dưới Llower với Pn là điểm bắt đầu  Bước 5: Bổ sung điểm thứ 3, nếu 3 điểm không tạo thành rẻ phải thì điểm giữa  Bước 6: Xoá điểm đầu và điểm cuối trong bao dưới Llower  Bước 7: Tập hợp điểm thu được theo chiều kim đồng hồ: Lupper U Llower  **Output:** Bao lồi (tập lồi nhỏ nhất chứa tất cả các điểm) |
| #include<bits/stdc++.h>  using namespace std;  typedef struct {  double x;  double y;  } Point;  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;  }  void convexhull(Point \*a, const int &n) {  Point t, Lup[n], Llow[n];    // Buble Sort  for(int i = 0; i < n; i++) {  for(int j = n - 2; 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)) {  t = a[j];  a[j] = a[j + 1];  a[j + 1] = t;  }  }  }    // Find Lupper  Lup[0] = a[0];  Lup[1] = a[1];  int j1 = 1;  for(int i = 2; i < n; i++) {  j1++;  Lup[j1] = a[i];  while(j1 >= 2 && determinant(Lup[j1 - 2], Lup[j1 - 1], Lup[j1]) > -0.0001) {  Lup[j1 - 1] = Lup[j1];  j1--;  }  }    // Find Llower  Llow[0] = a[n - 1];  Llow[1] = a[n - 2];  int j2 = 1;  for(int i = n - 3; i >= 0; i--) {  j2++;  Llow[j2] = a[i];  while(j2 >= 2 && determinant(Llow[j2 - 2], Llow[j2 - 1], Llow[j2]) > -0.0001) {  Llow[j2 - 1] = Llow[j2];  j2--;  }  }  // convex hull full  for(int i = 1; i <= j2 - 1; i++) {  Lup[j1 + i] = Llow[i];  }  cout<<"Bao loi: \n";  for(int i = 0; i < j1 + j2; i++) {  printf("(%lf, %lf)\n", Lup[i].x, Lup[i].y);  }  double s = 0;  int j = j1 + j2 - 1;  for (int i = 0; i < j1 + j2; i++)  {  s += (Lup[j].x + Lup[i].x) \* (Lup[j].y - Lup[i].y);  j = i;  }  s = abs(s) / 2;  cout<<"Dien tich bao loi: "<<s<<endl;;  }  int main() {  Point a[] = {  {6, 7},  {8, 6},  {9, 8},  {10, 9},  {11, 10},  {7, 12},  {6, 11},  {7, 11},  {10, 13},  {10, 7}  };  convexhull(a, sizeof(a)/sizeof(a[0]));  return 0;  }  Kết quả: |

1. **Bài toán cặp điểm gần nhất**

|  |
| --- |
| **Mô tả:** Từ tập hợp các điểm đề bài cho, ta cần tìm khoảng cách nhỏ nhất của 1 cặp điểm trong tất cả các điểm trên thoả mãn độ phức tạp của thuật toán là O(nlogn)  **Input:** Toạ độ các điểm theo đề bài  **Thuật toán:**  Bước 1: Sắp xếp tất cả các điểm theo toạ độ x  Bước 2: Chia tập hợp điểm thành 2 tập con có kích thước bằng nhau theo đường thẳng L  Bước 3: Giải bài toán bằng đệ quy trong tập con trái và tập con phải để tìm dleft và dright (khoảng cách cặp điểm nhỏ nhất của tập bên trái và tập bên phải)  Khi đó: d = min {dleft, dright};  Bước 4: Xây dựng miền S là tập hợp các điểm lân cận L  Bước 5: Tìm khoảng cách tối thiểu ds trong miền S (một điểm nằm bên trái và một điểm nằm bên phải L)  Bước 6: Kết luận khoảng cách cặp điểm gần nhất là d = min {d, ds}  **Output:** Khoảng cách của cặp điểm gần nhất |
| Code:  #include<bits/stdc++.h>  using namespace std;    struct Point  {  int x, y;  };    int compareX (const void\* a, const void\* b) {  Point \*p1 = (Point \*)a, \*p2 = (Point \*)b;  return (p1->x != p2->x) ? (p1->x - p2->x) : (p1->y - p2->y);  }  int compareY (const void\* a, const void\* b) {  Point \*p1 = (Point \*)a, \*p2 = (Point \*)b;  return (p1->y != p2->y) ? (p1->y - p2->y) : (p1->x - p2->x);  }    float dist (Point p1, Point p2) {  return sqrt ((p1.x - p2.x) \* (p1.x - p2.x) + (p1.y - p2.y) \* (p1.y - p2.y));  }    float bruteForce (Point P[], int n) {  float min = FLT\_MAX;  for (int i = 0; i < n; ++i)  for (int j = i + 1; j < n; ++j)  if (dist(P[i], P[j]) < min)  min = dist(P[i], P[j]);  return min;  }    float min (float x, float y) {  return (x < y)? x : y;  }    float stripClosest (Point strip[], int size, float d) {  float min = d;  for (int i = 0; i < size; ++i)  for (int j = i + 1; j < size && (strip[j].y - strip[i].y) < min; ++j)  if (dist(strip[i], strip[j]) < min)  min = dist(strip[i], strip[j]);    return min;  }  // Ham de quy tim khoang cach nho nhat  // Px [] chua cac diem sx theo toa do x  // Py [] chua cac diem sx theo toa do y  float closestUtil (Point Px[], Point Py[], int n)  {  if (n <= 3)  return bruteForce (Px, n);    int mid = n/2;  Point midPoint = Px[mid];      Point Pyl[mid]; // Pyl [] chua cac diem ben trai L  Point Pyr[n-mid]; // Pyr [] chua cac diem ben phai L  int li = 0, ri = 0;  for (int i = 0; i < n; i++)  {  if ((Py[i].x < midPoint.x || (Py[i].x == midPoint.x && Py[i].y < midPoint.y)) && li < mid)  Pyl[li++] = Py[i];  else  Pyr[ri++] = Py[i];  }    float dl = closestUtil (Px, Pyl, mid);  float dr = closestUtil (Px + mid, Pyr, n-mid);    float d = min (dl, dr);    Point strip[n];  int j = 0;  for (int i = 0; i < n; i++)  if (abs(Py[i].x - midPoint.x) < d)  strip[j] = Py[i], j++;    return stripClosest (strip, j, d);  }  float closest (Point P[], int n)  {  Point Px[n];  Point Py[n];  for (int i = 0; i < n; i++)  {  Px[i] = P[i];  Py[i] = P[i];  }    qsort (Px, n, sizeof(Point), compareX);  qsort (Py, n, sizeof(Point), compareY);    return closestUtil (Px, Py, n);  }    int main ()  {  Point P[] = {{1, 2}, {2, 5}, {3, 4}, {4, 3}, {5, 4}, {6, 1}, {7, 5}};  int n = sizeof(P) / sizeof(P[0]);  cout<<"n = "<<n<<endl;  cout << "Khoang cach nho nhat la " << closest (P, n);  return 0;  } |
| **Kết quả:** |

# **CHƯƠNG 4: OPTIMIZATIONS**

## **Bài toán 1: Gradient descent cho hàm 1 biến**

|  |
| --- |
| **Đề bài:** Tìm giá trị nhỏ nhất của hàm số  (Trong slide) |
| **Mô tả:** Tìm giá trị nhỏ nhất khó khăn do khó khăn trong việc tính đạo hàm. Dùng gradient descent, bắt đầu từ điểm  và cập nhật   * Điều kiện dừng của Gradient descent:   + Giới hạn số bước lặp  + So sánh giá trị hàm của nghiệm tại 2 lần cập nhật  + Kiểm tra giá trị tuyệt đối của gradient   * Tốc độ hội tụ của Gradient descent:   + Phụ thuộc vào điểm khởi tạo  + Phụ thuộc vào chỉ số alpha  **Input:** Hàm số  **Thuật toán:**   * Dự đoán một điểm khởi tạo * Cập nhật  đến đạt kết quả chấp nhận:   **Output:** |
| **Code:**  #include<bits/stdc++.h>  using namespace std;  double func(double x) {  return x \* x + 5 \* sin(x);  }  double grad(double x) {  return 2 \* x + 5 \* cos(x);  }  double gradientDescent(double x, double alpha, double gra, int loop) {  for(int i = 1; i <= loop; i++) {  x = x - alpha \* grad(x);  // cout<<"Buoc lap "<<i<<": x = "<<x<<", y = "<<func(x)<<endl;  if(abs(grad(x)) < gra)  break;  }  return x;  }  int main() {  double x0, alpha, gra;  int loop;  x0= 0;  alpha = 0.1;  gra = 0.001;  loop = 1000;  double x\_min = gradientDescent(x0, alpha, gra, loop);  double y\_min = func(x\_min);  printf("Diem khoi tao: x0 = %lf, y0 = %lf\n", x0, func(x0));  printf("Ham so dat nho nhat tai x = %lf, y = %lf", x\_min, y\_min);  return 0;  } |
| **Kết quả:** |

1. **Bài toán 2: Gradient descent cho hàm nhiều biến**

|  |
| --- |
| **Mô tả:** Cho hàm số  . Hãy tìm ra một điểm cực trị của nó. |
| **Input:** Hàm số cần tìm và các điểm khởi tạo  **Thuật toán:**  **Bước 0:** Bình thường hóa dữ liệu nếu cần thiết.  **Bước 1:** Chọn một điểm bất kỳ và một giá trị learning rate α.  **Bước 2:** Liên tiếp lặp lại các phép biến đổi    *…*    **Bước 3:** Thuật toán dừng lại khi y thay đổi rất nhỏ hoặc trị tuyệt đối các đạo hàm riêng rất nhỏ. Nếu thuật toán không thể kết thúc thì chọn giá trị α nhỏ hơn rồi quay lại bước 2  **Output:** Giá trị nhỏ nhất cục bộ của hàm số |
| **Code:**    #include<bits/stdc++.h>  using namespace std;  // Cho ham f(x) = x1^2 + x2^2  double func(double x1, double x2) {  return x1 \* x1 + x2 \* x2;  }  double grad1(double x) {  return 2 \* x;  }  double grad2(double x) {  return 2 \* x;  }  void gradientDescent(double x1, double x2, double alpha, double gra, int loop) {  double x1\_new, x2\_new;  for(int i = 1; i <= loop; i++) {  x1\_new = x1 - alpha \* grad1(x1);  x2\_new = x2 - alpha \* grad2(x2);  if(abs(x1\_new - x1) < gra && abs(x2\_new - x2) < gra)  break;  x1 = x1\_new;  x2 = x2\_new;  }  cout<<"Ham so dat gia tri nho nhat tai x1 = "<<x1<<", x2 = "<<x2<<endl;  cout<<"Gia tri nho nhat = "<<func(x1, x2);  }  int main() {  double x1, x2, alpha, gra;  int loop;  x1= -1;  x2 = 1;  alpha = 0.1;  gra = 0.00001;  loop = 1000;  gradientDescent(x1, x2, alpha, gra, loop);  return 0;  } |
| **Kết quả:** |

## **Bài toán 3: Gradient descent momentum cho hàm 1 biến**

***Đề bài 1:*** Cho hàm số .

|  |
| --- |
| **# Trả lời:** Khai triển đạo hàm: |

1. *(1 điểm) Khai triển đạo hàm cấp 1 của f(x)*
2. *(2 điểm)* Viết chương trình (có dùng hàm) tính giá trị bé nhất của f(x) sử dụng phương pháp *gradient descent với momentum* với tham số học (learning rate) , hệ số động lượng là , số bước lặp *N* và sai số :

|  |
| --- |
| **# Trả lời**: Dán code vào bên dưới:  #include <bits/stdc++.h>  #define gamma 0.01  #define N 1000  #define alpha 0.9  #define e 0.00001  using namespace std;  double function(double x) {  return (exp(x) + 3 \* x - 10) \* (exp(x) + 3 \* x - 10) + x \* x;  }  double grad(double x)  {  return 2 \* (exp(x) + 3 \* x - 10) \* (exp(x) + 3) + 2 \* x;  }  double GradientDescentWithMomentum(double x)  {  double v = 0;  double x\_new = 0;  for(int i = 1; i < N; i++)  {  // gamma: step\_size: learning rate  // alpha: momentum  v = alpha \* v + gamma \* grad(x);  x\_new = x - v;  cout<<"v = "<<v<<"; x = "<<x\_new<<endl;  if(abs(x\_new - x) < e)  return x\_new;  x = x\_new;  }  return x;  }  int main()  {  double x = 1.5;  double minimum = GradientDescentWithMomentum(x);  cout << "X\_min = "<<minimum << " ";// <<iterations;  cout << "Y\_min = "<<function(minimum);  return 0;  }  **# Trả lời**: Dán kết quả thực thi với điểm khởi , tham số học(*learning rate*) , hệ số động lượng (*momentum coefficient*) là , số bước lặp *N = 1000* và sai số : |

1. **Bài toán 4: Gradient descent with momentum với hàm nhiều biến**

***Đề bài 2:*** Trình bày phương pháp tối ưu Grandient Descent with momentum cho bài toán hồi quy tuyến tính hàm nhiều biến

1. *Phát biểu bài toán hồi quy tuyến tính và trình bày thuật toán giải*

|  |
| --- |
| **# Trả lời:** trình bày bài toán hồi quy tuyến tính ở đây:  Để tìm nghiệm tối ưu toàn cục và cục bộ của hàm số thì ta thường rất khó khăn hoặc không thể, khó khăn trong các bài toán với dữ liệu lớn hoặc chiều dữ liệu lớn do đó ta cần tìm 1 hướng tiếp cận để giải quyết vấn đề trên 1 cách dễ hơn.  Với bài toán hồi quy tuyến tính thì xuất phát từ một điểm và dùng phương pháp lặp để dịch chuyển đến nghiệm cần tìm  # **Trả lời**: Trình bày thuật toán giải bằng Grandient Descent with momentum  **Bước 1:** Dự đoán một điểm khởi tạo và vận tốc ban đầu:  **Bước 2:** Cập nhật đến khi đạt được kết quả chấp nhận:  Với  Trong đó  là vận tốc vật lý  thông tin về đà  thông tin về độ dốc |

1. Hãy viết chương trình giảỉ bằng Grandient Descent with momentum với tập dữ liệu minh họa

|  |
| --- |
| **# Trả lời**: viết câu trả lời vào bên dưới:  #include <bits/stdc++.h>  #define gamma 0.01  #define N 1000  #define alpha 0.9  #define e 0.00001  using namespace std;  // Cho ham so f(x) = x^1 + x^2  double function(double x1, double x2) {  return x1 \* x1 + x2 \* x2;  }  double grad1(double x1) {  return 2 \* x1;  }  double grad2(double x2)  {  return 2 \* x2;  }  void GradientDescentWithMomentum(double x1, double x2)  {  double v1 = 0, v2 = 0;  double x1\_new, x2\_new;  for(int i = 1; i < N; i++)  {  v1 = alpha \* v1 + gamma \* grad1(x1);  v2 = alpha \* v2 + gamma \* grad2(x2);  x1\_new = x1 - v1;  x2\_new = x2 - v2;    cout<<x1\_new<<" "<<x2\_new<<endl;  if(abs(x1\_new - x1) < e && abs(x2\_new - x2) < e)  break;  x1 = x1\_new;  x2 = x2\_new;  }  cout<<"Ham so dat nho nhat tai x1 = "<<x1 <<", x2 = "<<x2;  cout<<"\nMin = "<<function(x1, x2);  }  int main()  {  double x1 = 1, x2 = -1;  GradientDescentWithMomentum(x1, x2);  return 0;  }  **# Trả lời:** Dán kết quả minh họa với tập dữ liệu đa biến có sẳn |

1. **Phương pháp đơn hình**

|  |
| --- |
| **Mô tả:** |
| **Thuật toán:**  ***Bước 1:*** *Kiểm tra tính tối ưu*  *xo = (b1, b2, …, bm, 0, …, 0)*   * + Nếu mọi j=1...n: ∆j ≤ 0 thì xo là phương án tối ưu và   fmin= f(xo) = b1c1 + …. + bmcm   * + Nếu tồn tại k: *∆*k>0 thì chuyển sang bước 2.   ***Bước 2:*** *Kiểm tra điều kiện vô nghiệm*   * + Nếu tồn tại k: ∆k >0 và với mọi i = 1...m: aik≤0 thì **bài toán vô nghiệm.**   + *Nếu ∆k >0, và tồn tại i: aik>0* thì chuyển sang bước 3.   ***Bước 3:*** *Tìm ẩn thay thế và ẩn loại ra*   * + Nếu *∆*s = max {*∆*j} với *∆*j>0 (j=1…n) thì đưa **xs đưa vào tập ẩn cơ bản .**   + Nếu *b*r / ars = min {*b*i / ais} với ais> 0 thì **loại xr ra khỏi tập ẩn cơ bản .** Chuyển sang bước 4.   ***Bước 4:*** *Biến đổi bảng đơn hình*   * + Biến đổi bảng đơn hình: Tính lại các giá trị ∆j, f(x), quay lại *bước 1*. |
| **Code:**  #include<bits/stdc++.h>  using namespace std;  vector<int> HeSo(vector<int> c, vector< vector<double>> a){  vector<int> cb;  set<pair<int, int>> sp; //dung de tim he so theo thu tu hang 1 -> hang m      for(int j=0; j<a[0].size(); j++){    int ii= -1;  for(int i=0; i<a.size(); i++){  if (a[i][j] == 1 && ii == -1){  ii= i;  }  else  if (a[i][j]) ii= -2; //co the them break de giam dpt  }  if (ii>=0) sp.insert({ii, j}); //neu la 1 cot trong ma tran don vi  }    for(auto p : sp)  cb.push\_back(c[p.second]);  return cb;  }  vector<double> TinhDelta(vector<int> cb, vector<int> c, vector< vector<double>> a){  //deltai = sum(aij \* cbi) - cj  vector<double> v;    for(int j=0; j<a[0].size(); j++){  double s= 0;  for(int i=0; i<a.size(); i++)  s+= a[i][j] \* cb[i];  v.push\_back(s-c[j]);  }  return v;  }  double TinhFx(vector<int> cb, vector<double> b){  double s= 0;  int i= 0;  for(auto val\_b: b)  s += val\_b \* cb[i++];  return s;  }  bool VoNghiem(vector<double>delta, vector< vector<double>> a){  //neu ton tai deltaj > 0 ma aij <= 0 (voi moi i=1:m) thi vn    for(int j=0; j<delta.size(); j++)  if (delta[j] > 0){    bool check= true;  for(int i=0; i<a.size(); i++)  if (a[i][j] > 0) check= false;    if (check) return true;  }  return false;  }  bool PATU(vector<double>delta){  for(auto val:delta)  if (val > 0) return false;  return true;  }  vector<double> Nghiem(int m, vector< vector<double>> a, vector<double> b){  vector<double> x;    for(int j=0; j<a[0].size(); j++){    int ii= -1;  for(int i=0; i<a.size(); i++){  if (a[i][j] == 1 && ii == -1){  ii= i;  }  else  if (a[i][j]) ii= -2; //co the them break de giam dpt  }  if (ii>=0) x.push\_back(b[ii]); //neu la 1 cot trong ma tran don vi  else x.push\_back(0);  }    return x;  }  int TimCotVao(vector<double>delta){  double valMax= delta[0];  int index= 0;    for(int i=1; i<delta.size(); i++){  if (valMax < delta[i]){  valMax= delta[i];  index= i;  }  }  return index;  }  int TimCotRa(int jVao, vector<vector<double>>a, vector<double> b, int &iVao){  double valMin= (a[0][jVao]<=0 ? 100111000 : b[0]\*1.0 / a[0][jVao]);    for(int i=1; i<a.size(); i++){    if (a[i][jVao] <= 0) continue;  double tiSoKoAm= b[i]\*1.0 / a[i][jVao];      if (valMin > tiSoKoAm){    valMin= tiSoKoAm;  iVao= i;  }  }    for(int j=0; j<a[0].size(); j++)  if (j != jVao){    int ii= -1;  for(int i=0; i<a.size(); i++){  if (a[i][j] == 1 && ii == -1){  ii= i;  }  else  if (a[i][j]) ii= -2; //co the them break de giam dpt  }  if (ii == iVao) return j;  }  return -1; //error  }  void output(vector<int> c, vector<int> cb, vector<vector<double>> a, vector<double> delta, double fx, vector<double> b){  cout<<"\n\n-----------------------BANG DON HINH--------------------------------\n\n";  cout<<setw(10)<<""; for(auto val: c) cout<<setw(10)<<val; cout<<setw(10)<<" <-- [c]";  cout<<"\n";  int i= 0;  for(auto v : a){  cout<<"\n";    cout<<setw(10)<<cb[i];    for(auto val : v)  cout<<setw(10)<<fixed<<setprecision(3)<<val;    cout<<setw(10)<<b[i++];    }  cout<<"\n";  cout<<setw(10)<<"[cb]";  cout<<"\n";  cout <<setw(10)<<"[delta"; for(auto val:delta) cout<<setw(10)<<fixed<<setprecision(3)<<val; cout<<"]"<<setw(10)<<fixed<<setprecision(3)<<fx<<"=fx\n";  cout<<"\n\n-------------------------------------------------------------------------\n\n";  }  vector<double> process(int n, vector<int> c, int m, vector< vector<double>> a, vector<double> b){    vector<int> cb= HeSo(c, a);  vector<double> delta = TinhDelta(cb, c, a);  double fx = TinhFx(cb, b);    int cnt= 0;  while(true){  if (VoNghiem(delta, a)){  cout<<"Bai toan vo nghiem!";  exit(0);  }  if (PATU(delta)) return Nghiem(m, a, b);    int jv= TimCotVao(delta);  int iv= 0;  int jr= TimCotRa(jv, a, b, iv);    //bien doi ma tran a va vector b  double soNhan= 1 / a[iv][jv];  for(int j=0; j<a[0].size(); j++) a[iv][j] \*= soNhan;  b[iv] \*= soNhan;    for(int i=0; i<a.size(); i++)  if (i != iv){    soNhan = -a[i][jv] / a[iv][jv];  for(int j=0; j<a[i].size(); j++)  a[i][j] += a[iv][j]\*soNhan;    b[i] += b[iv] \* soNhan;  }    //bien doi vector delta  soNhan= -delta[jv] / a[iv][jv];  for(int j= 0; j<a[0].size(); j++)  delta[j] += a[iv][j] \* soNhan;  fx += b[iv] \* soNhan;    output(c, cb, a, delta, fx, b);  }  return vector<double>();  }    void input(){  //freopen("input.txt","r",stdin);  //cout<<"nhap n (so an x (co so + ko co so)) : ";  int n; cin >> n;    vector<int> c(n);  for(int i=0; i<n; i++){  //cout<<"nhap c"<<i<<" : ";  cin >> c[i];  }    //cout<<"nhap m (so an x (co so)): ";  int m; cin >> m;    vector< vector<double> > a(m, vector<double>(n));  vector<double> b(m);    for(int i=0; i<a.size(); i++){  for(int j=0; j<a[i].size(); j++){    //cout<<"nhap a"<<i<<j<<" : ";  cin >> a[i][j];  }  //cout<<" sum(Aij \* xj) = bi, nhap b"<<i<<" : ";  cin >> b[i];  }    vector<double> x= process(n, c, m, a, b);  cout<<"VAY NGHIEM TOI UU LAN LUOT LA: \n";  for(int i=0; i<x.size(); i++)  cout<<"x"<<i<<" = "<<x[i]<<"\n";  }  int main(){    input();    } |
| Kết quả: |

1. **Phương pháp đơn hình 2 pha**

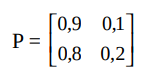
|  |
| --- |
| **Mô tả:** |
| **Thuật toán:** |
| **Code:**  #include<bits/stdc++.h>  using namespace std;  vector<int> v;  int soAnThiet;  int pha= 0;    vector<int> HeSo(vector<int> c, vector< vector<double>> a){      vector<int> cb;      set<pair<int, int>> sp;             //dung de tim he so theo thu tu hang 1 -> hang m          for(int j=0; j<a[0].size(); j++){            int ii= -1;          for(int i=0; i<a.size(); i++){              if (a[i][j] == 1 && ii == -1){                  ii= i;              }              else                  if (a[i][j]) ii= -2;    //co the them break de giam dpt          }          if (ii>=0) sp.insert({ii, j});      //neu la 1 cot trong ma tran don vi      }        for(auto p : sp)          cb.push\_back(c[p.second]);      return cb;  }  vector<double> TinhDelta(vector<int> cb, vector<int> c, vector< vector<double>> a){      //deltai = sum(aij \* cbi) - cj      vector<double> v;        for(int j=0; j<a[0].size(); j++){          double s= 0;          for(int i=0; i<a.size(); i++)              s+= a[i][j] \* cb[i];          v.push\_back(s-c[j]);      }      return v;  }  double TinhFx(vector<int> cb, vector<double> b){      double s= 0;      int i= 0;      for(auto val\_b: b)          s += val\_b \* cb[i++];      return s;  }  bool VoNghiem(vector<double>delta, vector< vector<double>> a){      //neu ton tai deltaj > 0 ma aij <= 0 (voi moi i=1:m) thi vn        for(int j=0; j<delta.size(); j++)      if (delta[j] > 0){            bool check= true;          for(int i=0; i<a.size(); i++)              if (a[i][j] > 0) check= false;            if (check) return true;      }      return false;  }  bool PATU(vector<double>delta){      for(auto val:delta)          if (val > 0) return false;      return true;  }  vector<double> Nghiem(vector< vector<double>> a, vector<double> b){      vector<double> x;        for(int j=0; j<a[0].size(); j++){            int ii= -1;          for(int i=0; i<a.size(); i++){              if (a[i][j] == 1 && ii == -1){                  ii= i;              }              else                  if (a[i][j]) ii= -2;    //co the them break de giam dpt          }          if (ii>=0) x.push\_back(b[ii]);  //neu la 1 cot trong ma tran don vi              else x.push\_back(0);      }        return x;  }  int TimCotVao(vector<double>delta){      double valMax= delta[0];      int index= 0;        for(int i=1; i<delta.size(); i++){          if (valMax < delta[i]){              valMax= delta[i];              index= i;          }      }      return index;  }  int TimCotRa(int jVao, vector<vector<double>>a, vector<double> b, int &iVao){      double valMin= (a[0][jVao]<=0 ? 100111000 : b[0]\*1.0 / a[0][jVao]);        for(int i=1; i<a.size(); i++){            if (a[i][jVao] <= 0) continue;          double tiSoKoAm= b[i]\*1.0 / a[i][jVao];              if (valMin > tiSoKoAm){                valMin= tiSoKoAm;              iVao= i;          }      }        for(int j=0; j<a[0].size(); j++)      if (j != jVao){            int ii= -1;          for(int i=0; i<a.size(); i++){              if (a[i][j] == 1 && ii == -1){                  ii= i;              }              else                  if (a[i][j]) ii= -2;    //co the them break de giam dpt          }          if (ii == iVao) return j;      }      return -1;      //error  }  void output(vector<int> c, vector<int> cb, vector<vector<double>> a, vector<double> delta, double fx, vector<double> b){      cout<<"\n\n-----------------------BANG DON HINH--------------------------------\n\n";      cout<<setw(10)<<""; for(auto val: c) cout<<setw(10)<<val; cout<<setw(10)<<" <-- [c]";      cout<<"\n";      int i= 0;      for(auto v : a){          cout<<"\n";            cout<<setw(10)<<cb[i];            for(auto val : v)              cout<<setw(10)<<fixed<<setprecision(3)<<val;            cout<<setw(10)<<b[i++];        }      cout<<"\n";      cout<<setw(10)<<"[cb]";      cout<<"\n";      cout <<setw(10)<<"[delta"; for(auto val:delta) cout<<setw(10)<<fixed<<setprecision(3)<<val; cout<<"]"<<setw(10)<<fixed<<setprecision(3)<<fx<<"=fx\n";      cout<<"\n\n-------------------------------------------------------------------------\n\n";  }  vector< vector<double> > XoaCotGia(vector< vector<double> > a){      vector< vector<double> > res;        for(auto v:a){          vector<double> vd;          for(int i=0; i<soAnThiet; i++)              vd.push\_back(v[i]);          res.push\_back(vd);      }  return res;  }  vector<double> process(int n, vector<int> c, int m, vector< vector<double>> a, vector<double> b){      cout<<"\n\n\n\n@@@@@@@@@@@@@@@ PHA "<<++pha<<" @@@@@@@@@@@@@@@\n\n\n\n";        vector<int> cb= HeSo(c, a);      vector<double> delta = TinhDelta(cb, c, a);      double fx = TinhFx(cb, b);        int cnt= 0;      while(true){            output(c, cb, a, delta, fx, b);          if (VoNghiem(delta, a)){              cout<<"Bai toan vo nghiem!";              exit(0);          }          if (PATU(delta)){              if (pha == 1){                  vector<double> nghiem = Nghiem(a, b);                    double s= 0;                  for(int i=soAnThiet; i<nghiem.size(); i++)                      s += nghiem[i];                  if (s > 0.00000001){                      cout<<"Vo nghiem do g(t\*) > 0";                      exit(0);                  }              }              return (pha == 1 ? process(soAnThiet, v, m, XoaCotGia(a), b) : Nghiem(a, b));          }          int jv= TimCotVao(delta);          int iv= 0;          int jr= TimCotRa(jv, a, b, iv);            //bien doi ma tran a va vector b          double soNhan= 1 / a[iv][jv];          for(int j=0; j<a[0].size(); j++) a[iv][j] \*= soNhan;          b[iv] \*= soNhan;            for(int i=0; i<a.size(); i++)          if (i != iv){                soNhan = -a[i][jv] / a[iv][jv];              for(int j=0; j<a[i].size(); j++)                  a[i][j] += a[iv][j]\*soNhan;                b[i] += b[iv] \* soNhan;          }            //bien doi vector delta          soNhan= -delta[jv] / a[iv][jv];          for(int j= 0; j<a[0].size(); j++)              delta[j] += a[iv][j] \* soNhan;          fx += b[iv] \* soNhan;          }      return vector<double>();  }      void input(){      freopen("donhinh2pha.txt","r",stdin);      //cout<<"nhap n (so an x (co so + ko co so)) : ";      int n;  cin >> n;        vector<int> c(n);      for(int i=0; i<n; i++){          //cout<<"nhap c"<<i<<" : ";          cin >> c[i];      }        //cout<<"nhap m (so an x (co so)): ";      int m;  cin >> m;        vector< vector<double> > a(m, vector<double>(n));      vector<double> b(m);        for(int i=0; i<a.size(); i++){          for(int j=0; j<a[i].size(); j++){                //cout<<"nhap a"<<i<<j<<" : ";              cin >> a[i][j];          }          //cout<<" sum(Aij \* xj) = bi, nhap b"<<i<<" : ";          cin >> b[i];      }    //cout<<"Nhap so an thiet:";  cin >> soAnThiet;  for(int i=0; i<soAnThiet; i++){  int x;  cin >> x;  v.push\_back(x);  }      vector<double> x= process(n, c, m, a, b);      cout<<"VAY NGHIEM TOI UU LAN LUOT LA: \n";      for(int i=0; i<x.size(); i++)          cout<<"x"<<i<<" = "<<x[i]<<"\n";  }  int main(){        input();    } |
| **Kết quả:** |

# **CHƯƠNG 5: PROBABILITY**

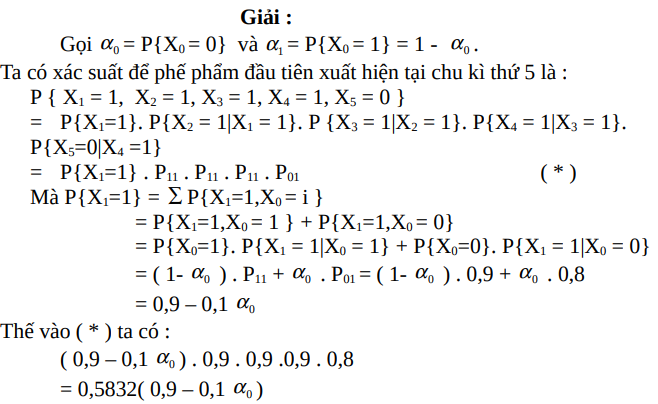
## **Chuỗi markov**

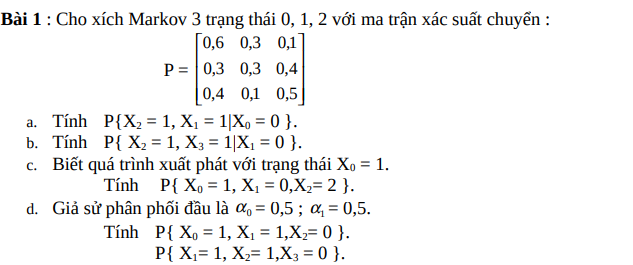
+ Kết quả tương lai chỉ dựa vào trạng thái hiện tại

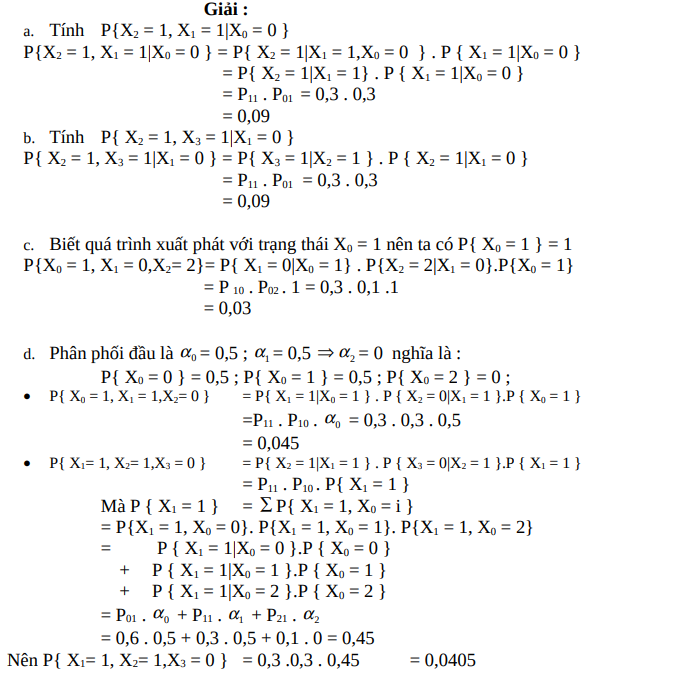
**Bài tập minh hoạ:** Giả sử 1 hệ thống sản xuất có chất lượng sản phẩm tuân theo xích Markov với kí hiệu 1 cho sản phẩm tốt và 0 cho sản phẩm xấu và ma trận xác suất chuyển là:

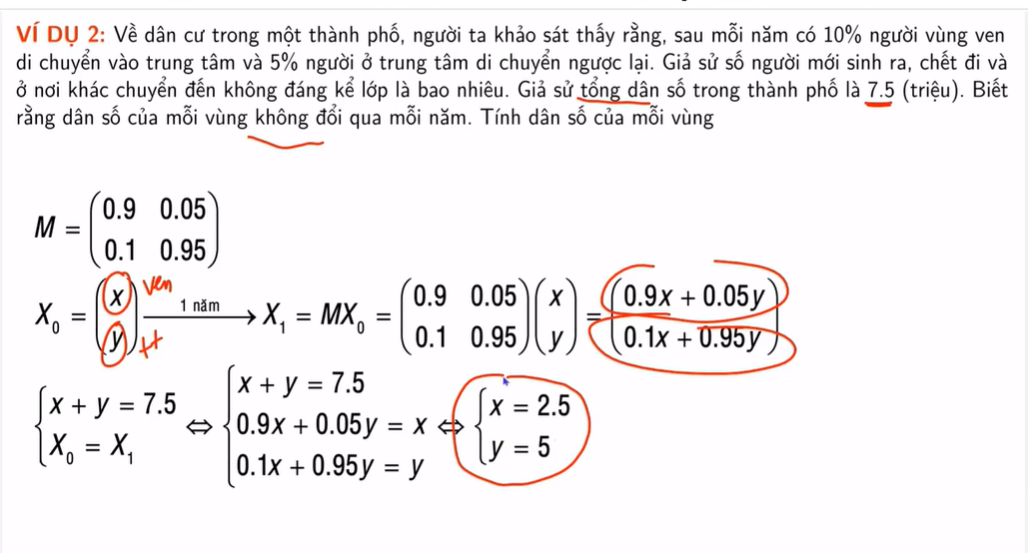


Tính xác suất để phế phẩm đầu tiên xuất hiện tại chu kì thứ 5.









***Câu 5*** (*1 điểm*): Một hệ thống có chế độ làm việc ở mỗi giai đoạn vận hành chỉ với hai trạng thái 0 và 1. Chế độ làm việc của hệ thống này được mô tả bằng chuỗi Markov như hình vẽ.

ii

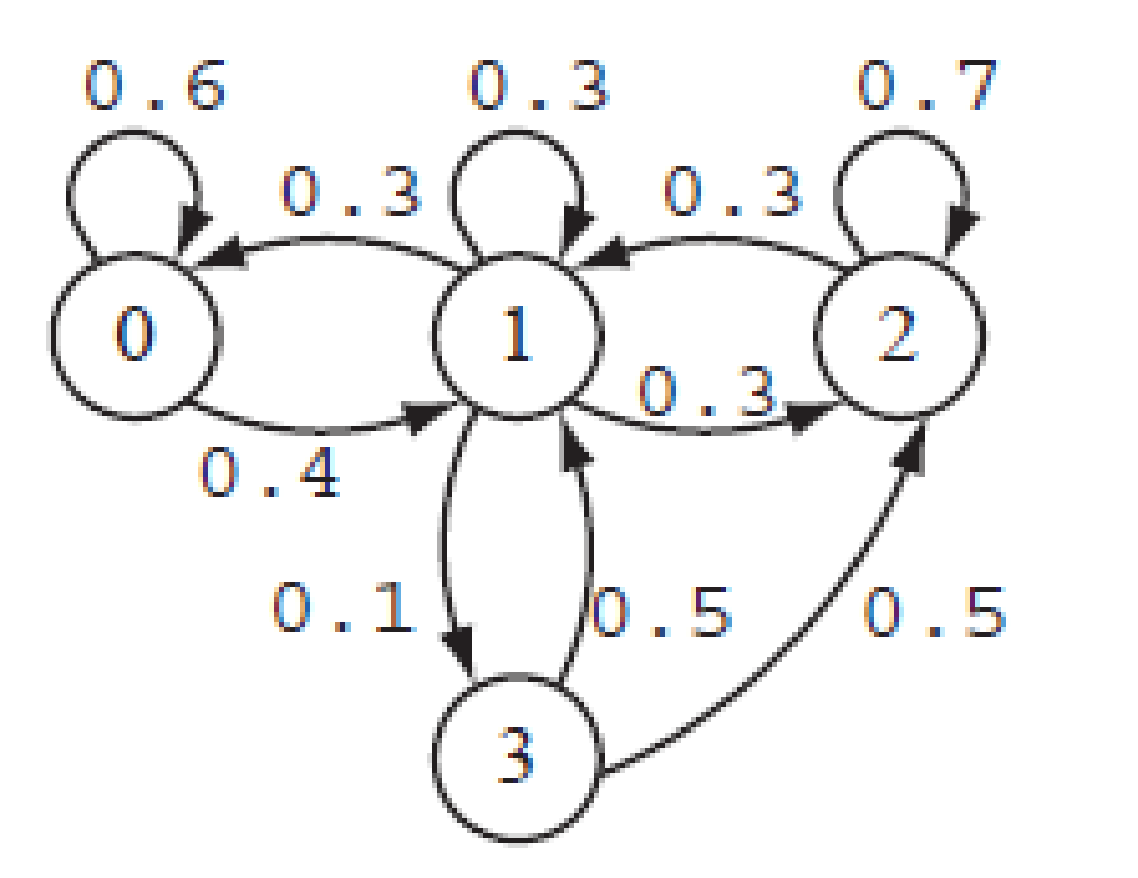
a) (*0.5 điểm*) Xác định ma trận chuyển đổi trạng thái **P** của hệ.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **# Trả lời:** dán kết quả vào bên dưới:   |  |  |  | | --- | --- | --- | | Giai đoạn vận hành hiện tại | Giai đoạn vận hành tiếp theo | | | Trạng thái 0 | Trạng thái 1 | | Trạng thái 0 | 0.22 | 0.78 | | Trạng thái 1 | 0.72 | 0.28 |   Ma trận chuyển đổi trạng thái P của hệ là: |

b) (*0.5 điểm*) Tìm xác suất (lớn nhất) khi hệ thống vẫn làm việc ở trạng thái **0** sau hai giai đoạn vận hành biết rằng hệ thống bắt đầu làm việc ở trạng thái **0**.

|  |
| --- |
| **# Trả lời**: Dán kết quả tính toán vào bên dưới:  Gọi π (i) là vectơ xác suất trạng thái cho giai đoạn vận hành i  Ta có giai đoạn vận hành 0 trong trường hợp hệ thống bắt đầu làm việc ở trạng thái 0  π (0) = (1,0)  Xác suất trạng thái tại giai đoạn vận hành thứ 1  π (1) = π (0) \* P = (1 0) \* = (0.22 0.78)  Xác suất trạng thái tại giai đoạn vận hành thứ 2  π (2) = π (1) \* P = (0.22 0.78) \* = (0.61 0.39)  Vậy xác suất lớn nhất khi hệ thống vẫn làm việc ở trạng thái 0 sau hai giai đoạn vận hành biết rằng hệ thống bắt đầu làm việc ở trạng thái 0 là 61% |
| **Code:**  #include <iostream>  #include <iomanip>  using namespace std;  #define R 2 // So hang ma tran chuyen  #define C 2 // So cot ma tran chuyen  // Nhan 2 ma tran  double\*\* mulMat(double \*\*m1, double \*\*m2, int row1, int col1, int row2, int col2) {  double\*\* res;  res = new double \*[R];  for (int i = 0; i < row1; i++) {  res[i] = new double[C];  for (int j = 0; j < col2; j++) {  res[i][j] = 0;  for (int k = 0; k < col1; k++)  res[i][j] += m1[i][k] \* m2[k][j];  }  }  return res;  }  // Tim ma tran chuyen sau n giai doan  double\*\* powMat(double \*m[C], int step) {  double \*\*res = m;  while (--step) {  res = mulMat(res, m, R, C, R, C);  }  return res;  }  int main() {  // p: ma tran chuyen  double \*\*p = new double \*[R];  int t = 0;  cout<<"Nhap ma tran chuyen ban dau: \n";  for (int i = 0; i < R ; i++) {  p[i] = new double[C];  for (int j = 0; j < C; j++) {  cout<<"Tu "<<j<<" den "<<i<<" = ";  cin >> p[i][j];  }  }  // x: ma tran trang thai ban dau  cout<<"Nhap ma tran trang thai ban dau: \n";  double \*\*x = new double \*[R];  for (int i = 0; i < R ; i++) {  x[i] = new double[1];  for (int j = 0; j < 1; j++) {  cout<<"Xac suat o trang thai "<<i<<" = ";  cin>>x[i][j];  }  }    cout << "Nhap so giai doan = ";  cin >> t;  double \*\*res = powMat(p, t);    cout << "Ma tran chuyen sau " << t << " giai doan: " << endl;  for (int i = 0; i < R; i++)  {  for (int j = 0; j < C; j++)  cout << setw(15) << res[i][j];  cout << endl;  }  cout<< "Xac suat he thong lam viec sau "<< t <<" giai doan: "<<endl;  double \*\*result = mulMat(res, x, R, C, R, 1);  for (int i = 0; i < R; i++)  {  for (int j = 0; j < 1; j++)  cout << setw(15) << result[i][j];  cout << endl;  }  } |
| **Kết quả:** |

***Câu 5*** (*1 điểm*): Một hệ thống có chế độ làm việc ở mỗi giai đoạn vận hành chỉ với các trạng thái từ 0 đến 3. Chế độ làm việc của hệ thống này được mô tả bằng chuỗi Markov như hình vẽ.



a) (*0.5 điểm*) Xác định ma trận chuyển đổi trạng thái **P** của hệ.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **# Trả lời:** dán kết quả vào bên dưới:  **# Trả lời:** dán kết quả vào bên dưới:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | Giai đoạn vận hành hiện tại | Giai đoạn vận hành tiếp theo | | | | | Trạng thái 0 | Trạng thái 1 | Trạng thái 2 | Trạng thái 3 | | Trạng thái 0 | 0.6 | 0.4 | 0 | 0 | | Trạng thái 1 | 0.3 | 0.3 | 0.3 | 0.1 | | Trạng thái 2 | 0 | 0.3 | 0.7 | 0 | | Trạng thái 3 | 0 | 0.5 | 0.5 | 0 |   Ma trận chuyển đổi trạng thái P của hệ là: |

b) (*0.5 điểm*) Tìm xác suất (lớn nhất) khi hệ thống làm việc ở trạng thái **3** sau *ba giai đoạn vận hành* biết rằng hệ thống bắt đầu làm việc ở trạng thái **1**.

|  |
| --- |
| **# Trả lời**: Dán kết quả tính toán vào bên dưới:  Gọi π (i) là vectơ xác suất trạng thái cho giai đoạn vận hành i  Ta có ở giai đoạn vận hành 0 thì hệ thống làm việc ở trạng thái 1 nên ta có:  π (0) = [0 1 0 0]  Vecto xác suất trạng thái tại giai đoạn vận hành thứ 1 là:  π (1) = π (0) \* P = [0 1 0 0] \*  = [0.3 0.3 0.3 0.1]  Vecto xác suất trạng thái tại giai đoạn vận hành thứ 2 là:  π (2) = π (1) \* P = [0.3 0.3 0.3 0.1] \*  = [0.27 0.35 0.35 0.03]  Vecto xác suất trạng thái tại giai đoạn vận hành thứ 3 là:  π (3) = π (2) \* P = [0.27 0.35 0.35 0.03] \*  = [0.267 0.333 0.365 0.035]  Vậy xác suất lớn nhất khi hệ thống vẫn làm việc ở trạng thái 3 sau ba giai đoạn vận hành biết rằng hệ thống bắt đầu làm việc ở trạng thái 1 là 0.035 |