TRƯỜNG ĐẠI HỌC BÁCH KHOA

**KHOA CÔNG NGHỆ THÔNG TIN**

**ĐỀ THI VÀ BÀI LÀM**

Tên học phần: Toán ứng dụng CNTT
Mã học phần: Hình thức thi: *Tự luận có giám sát*
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:** …Lý Thanh Hải……

**Lớp**:……19TCLC_DT2……

**MSSV**:……102190061……...

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*):

a) *(1 điểm)* Cho số nguyên dương n (n>1). Hãy viết hàm phân tích n ra thành tích hữu hạn của các số nguyên tố.

**# Trả lời:** Dán code vào bên dưới:

```
#include <stdio.h>
#include <math.h>
#include <iostream>
using namespace std;

int main() {
    int n;
    int i;
    cout << "Nhap n = ";
    cin >> n;

    cout << "Ket qua khi phan tich thua so la: ";
    for (i = 2; i <= n; i++)
    {
        while(n % i == 0) {
            cout << i << ".";
            n /= i;
        }
    }
    cout << endl;
    return 0;
}
```

# **Trả lời:** Dán kết quả thực thi vào bên dưới (với n=22000):

```
d:\Class\ky1_2122\ToánUD\thiCK>cd "d:\Class\ky1_2122\ToánUD\thiCK\" && g++ bai1.cpp -o bai1 && "d:\Class\ky1_2122\ToánUD\thiCK\"bai1
Nhap n = 22000
Ket qua khi phan tich thua so la: 2.2.2.2.5.5.5.11.
```

b) *(1 điểm)* Cho hệ đồng dư sau: $\begin{cases} x \equiv 2 \bmod 3 \\ x \equiv 3 \bmod 5 \\ x \equiv 4 \bmod 7 \end{cases}$.

Viết hàm giải hệ phương trình đồng dư trên.

# **Trả lời:** Dán code vào bên dưới:

```cpp
#include <iostream>
#include <stdio.h>
using namespace std;

int GCD(int a, int b)
{
    if(b == 0)
    {
        return a;
    }
    return GCD(b, a % b);
}

int main(){
    int a = 2, b = 3;
    cout <<"uoc chung lon nhat la: " << GCD(a, b) << endl;
    int a1 = 3, b1 = 5;
    cout <<"uoc chung lon nhat la: " << GCD(a1, b1) << endl;
    int a2 = 4, b2 = 7;
    cout <<"uoc chung lon nhat la: " << GCD(a2, b2) << endl;
    return 0;
}
```

# **Trả lời:** Dán kết quả thực thi vào bên dưới:

```
d:\Class\ky1_2122\ToánUD\thiCK>cd "d:\Class\ky1_2122\ToánUD\thiCK\" && g++ bai11.cpp -o bai11 && "d:\Class\ky1_2122\ToánUD\thiCK\"bai11
uoc chung lon nhat la: 1
uoc chung lon nhat la: 1
uoc chung lon nhat la: 1
```

***Câu 2*** (*3 điểm*): Cho ma trận A. Viết hàm 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):

- Điều kiện để phân rã ma trận A bằng phương pháp SVD là:

+ A là ma trận vuông không suy biến.

+ A là ma trận thực.

+ A là mà trận vuông đối xứng không suy biến.

- Code

```
#include "../Eigen/Eigenvalues"
#include <iostream>
#include <complex>
using namespace std;

int main () {

    int n = 0, m = 0;
    cout << "nhap n =";
    cin >> n;
    cout << "nhap m =";
    cin >> m;

    Eigen::MatrixXd A(n, m);
    Eigen::MatrixXd B(m, n);
    Eigen::MatrixXd C(n, n);
    Eigen::MatrixXd D(m, m);

    // ma trận gốc
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            cout << "nhap a[" << i << "][" << j << "]: ";
            cin >> A(i,j);
        }
    }

    cout << "matrix A:" << endl << A << endl;

    // ma trận chuyển vị từ ma trận gốc
    for(int i = 0; i < n; i++)
    {
        for(int j = 0; j < m; j++)
        {
          B(j,i) = A(i,j);
        }
    }

    // nhân 2 ma trận B, A
    for(int i = 0; i < n; i++)
    {
```

```
        for(int j = 0; j < n; j++)
        {
            C(i,j) = 0;
            for(int l = 0; l < m; l++)
            {
                C(i,j) += B(l,j) * A(i,l);
            }
        }
    }

    // nhân 2 ma trận A, B
    for(int i = 0; i < m; i++)
    {
        for(int j = 0; j < m; j++)
        {
            D(i,j)=0;
            for(int l = 0; l < n; l++)
            {
              D(i,j) += B(i,l) * A(l,j);
            }
        }
    }

    Eigen::EigenSolver < Eigen::MatrixXd> s (C);
    Eigen::EigenSolver < Eigen::MatrixXd> t (D);
    Eigen::JacobiSVD<Eigen::MatrixXd> svd(A, Eigen::ComputeFullU |
Eigen::ComputeFullV);
    cout << "B:" << endl;
    cout << B << endl;

    cout << s.eigenvalues() << endl;
    cout << "D: " << endl << svd.singularValues() << endl;
    cout << "U:" << endl;
    cout << s.eigenvectors() << endl;
    cout << "V:"<<endl;
    cout << t.eigenvectors();
}
```

# **Trả lời:** Dán kết quả thực thi vào bên dưới với $A=\begin{bmatrix} 3 & -1 \\ -1 & 5 \end{bmatrix}$ (sai số $\varepsilon = 10^{-5}$)

```
d:\Class\ky1_2122\ToánUD\thiCK>cd "d:\Class\ky1_2122\ToánUD\thiCK\" && g++ bai2.cpp -o bai2 && "d:\Class\ky1_2122\ToánUD\thiCK\"bai2
nhap n = 2
nhap m = 2
nhap a[0][0]: 3
nhap a[0][1]: -1
nhap a[1][0]: -1
nhap a[1][1]: 5
matrix A:
 3 -1
-1  5
B:
 3 -1
-1  5
(6.68629,0)
(29.3137,0)
D:
5.41421
2.58579
U:
 (-0.92388,0)  (0.382683,0)
(-0.382683,0)  (-0.92388,0)
V:
 (-0.92388,0)  (0.382683,0)
(-0.382683,0)  (-0.92388,0)
```

***Câu 3*** (2 *điểm*): Cho mười điểm trong không gian Oxy như sau: (0, 1); (2,0); (3,2); (4,3); (5,4); (1,6); (0,5); (1,5); (4,7); (4,1)

a) *(1.0 điểm) Mô tả thuật toán xác định bao lồi và xác định bao lồi*

**# Trả lời:** dán sơ đồ khối hoặc ngôn ngữ tự nhiên vào bên dưới:

* Dùng thuật toán Graham để xác định bao lồi:

- Tìm điểm tận cùng bằng cách so sánh tọa độ y của tất cả các điểm. Nếu có hai điểm có cùng giá trị y thì điểm có giá trị tọa độ x nhỏ hơn là giá trị được xem xét. Gọi điểm cực đại là P0. Đặt P0 ở vị trí đầu tiên trong bao lồi đầu ra.

- Xét n-1 điểm còn lại và sắp xếp chúng theo góc cực theo thứ tự ngược chiều kim đồng hồ xung quanh các điểm 0. Nếu góc cực của hai điểm bằng nhau thì đặt điểm gần nhất trước.

- Sau khi sắp xếp, hãy kiểm tra xem hai hoặc nhiều điểm có cùng góc với nhau hay không. Nếu thêm hai điểm có cùng góc thì xóa tất cả các điểm cùng góc trừ điểm xa P0 nhất. Gọi kích thước của mảng mới là m.

- Nếu m nhỏ hơn 3, trả về không thể có bao lồi.

- Tạo một ngăn xếp rỗng Stack và đẩy điểm [0], điểm [1] và điểm [2] đến Stack.

- Xử lý từng điểm m - 3 còn lại. Thực hiện theo cho mọi điểm điểm [i].

- Tiếp tục xóa điểm khỏi ngăn xếp trong khi hướng của 3 điểm theo sau không ngược chiều kim đồng hồ, điểm cạnh trên cùng trong ngăn xếp, trỏ vào trên cùng của ngăn xếp, điểm [i].

- Thêm điểm [i] đến Stack.

- In các điểm của Stack trên có thể được chia thành hai giai đoạn.

+ Giai đoạn 1(Sắp xếp điểm): Đầu tiên chúng ta tìm điểm dưới cùng. Ý tưởng là để xử lý trước các điểm là sắp xếp chúng theo điểm cuối cùng. Khi các điểm được sắp xếp, chúng tạo thành một đường khép kín.

+ Giai đoạn 2(Chấp nhận hoặc Từ chối điểm): Khi chúng ta đã có đường dẫn đóng, bước tiếp theo là đi ngang đường dẫn và loại bỏ phần lõm trên con đường này. Hai điểm đầu tiên trong mảng đã sắp xếp luôn là một phần của bao lồi. Đối với các điểm còn lại, theo dõi ba điểm hiện tại và tìm góc tạo bởi chúng. Cho ba điểm là trước (p), hiện tại (c) và sau (n). Nếu hướng của các điểm này không ngược chiều kim đồng hồ, chúng ta loại bỏ c, nếu không chúng ta giữ nguyên.

b) *(1.0 điểm)* Viết hàm xác định bao lồi

# **Trả lời:** Dán code bên dưới:

```
#include <iostream>
#include <stack>
#include <stdlib.h>
using namespace std;

struct Point
{
    int x, y;
};

Point p0;

Point nextToTop(stack<Point> &S)
{
    Point p = S.top();
    S.pop();
    Point res = S.top();
    S.push(p);
    return res;
}

void swap(Point &p1, Point &p2)
{
    Point temp = p1;
    p1 = p2;
    p2 = temp;
}

int distSq(Point p1, Point p2)
{
    return (p1.x - p2.x)*(p1.x - p2.x) +
        (p1.y - p2.y)*(p1.y - p2.y);
}

int orientation(Point p, Point q, Point r)
{
    int val = (q.y - p.y) * (r.x - q.x) -
```

```
            (q.x - p.x) * (r.y - q.y);

    if (val == 0) return 0;
    return (val > 0)? 1: 2;
}


int compare(const void *vp1, const void *vp2)
{
    Point *p1 = (Point *)vp1;
    Point *p2 = (Point *)vp2;

    int o = orientation(p0, *p1, *p2);
    if (o == 0)
        return (distSq(p0, *p2) >= distSq(p0, *p1))? -1 : 1;

    return (o == 2)? -1: 1;
}

void convexHull(Point points[], int n)
{

    int ymin = points[0].y, min = 0;
    for (int i = 1; i < n; i++)
    {
        int y = points[i].y;
        if ((y < ymin) || (ymin == y &&
            points[i].x < points[min].x))
            ymin = points[i].y, min = i;
    }

    swap(points[0], points[min]);

    p0 = points[0];
    qsort(&points[1], n-1, sizeof(Point), compare);

    int m = 1;
    for (int i=1; i<n; i++)
    {
        while (i < n-1 && orientation(p0, points[i],
                                      points[i+1]) == 0)
            i++;


        points[m] = points[i];
        m++;
    }

    if (m < 3) return;

    stack<Point> S;
    S.push(points[0]);
```

```
    S.push(points[1]);
    S.push(points[2]);


    for (int i = 3; i < m; i++)
    {
        while (S.size()>1 && orientation(nextToTop(S), S.top(), points[i]) != 2)
            S.pop();
        S.push(points[i]);
    }

    while (!S.empty())
    {
        Point p = S.top();
        cout << "(" << p.x << ", " << p.y <<")" << endl;
        S.pop();
    }
}

int main()
{
    int m;
    cout << "Nhap m = ";
    cin >> m;

    Point points[m] = {};
    for (int i = 0; i < m; i++)
    {
        Point p;
        cout << "nhap x point[" << i << "] = ";
        cin >> p.x;
        cout << "nhap y point[" << i << "] = ";
        cin >> p.y;
        points[i] = p;
    }
    int n = sizeof(points)/sizeof(points[0]);
    cout << "Ket qua Bao loi: " << endl;
    convexHull(points, n);
    cout << endl;
    return 0;
}
```

# **Trả lời:** Dán kết quả thực thi vào bên dưới:

```
d:\Class\ky1_2122\ToánUD\thiCK>cd "d:\Class\ky1_2122\ToánUD\thiCK\" && g++ bai3.cpp -o bai3 && "d:\Class\ky1_2122\ToánUD\thiCK\"bai3
Nhap m = 10
nhap x point[0] = 0
nhap y point[0] = 1
nhap x point[1] = 2
nhap y point[1] = 0
nhap x point[2] = 3
nhap y point[2] = 2
nhap x point[3] = 4
nhap y point[3] = 3
nhap x point[4] = 5
nhap y point[4] = 4
nhap x point[5] = 1
nhap y point[5] = 6
nhap x point[6] = 0
nhap y point[6] = 5
nhap x point[7] = 1
nhap y point[7] = 5
nhap x point[8] = 4
nhap y point[8] = 7
nhap x point[9] = 4
nhap y point[9] = 1
Ket qua Bao loi:
(0, 1)
(0, 5)
(1, 6)
(4, 7)
(5, 4)
(4, 1)
(2, 0)
```

***Câu 4*** (2 *điểm*): Cho hàm số $f(x)=\left(\ln(x^2+10)+x-5\right)^2+2x$

a) *(1 điểm) Khai triển đạo hàm cấp 1 của f(x)*

# **Trả lời:** Khai triển đạo hàm:

Lời giải: Đạo hàm cấp 1.

$$f(x) = (\ln(x^2+10) + x - 5) + 2x$$

$$\Rightarrow f'(x) = ((\ln(x^2+10) + x - 5) + 2x)'$$

$$= \frac{2.(\ln(x^2+10) + x - 5).(x^2+2x+10)}{x^2+10} + 2$$

$$= \frac{(\ln(x^2+10) + x - 5).(2x^2+4x+20)}{x^2+10} + 2$$

b) *(1 điểm)* Viết chương trình (có dùng hàm) tính giá trị bé nhất của f(x) sử dụng phương pháp *gradient descent* với tham số học (learning rate) $\gamma$, số bước lặp $N$ và sai số $\varepsilon$.

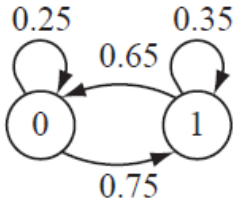
# **Trả lời**: Dán code vào bên dưới:

```
#include <iostream>
#include <math.h>
using namespace std;

// hàm f(x)
double cost(int x)
{
    return pow((log(pow(x, 2) + 10) + x - 5), 2) + 2 * x;
}

// đạo hàm f'(x)
double grad(int x)
{
    return (log(pow(x, 2) + 10) + x - 5) * (4 * x + 2 * pow(x, 2) + 20) / (pow(x, 2) +
10) + 2;
}

// hàm tính gradient_discent
double myGradient_Discent(double alpha, int x0, double gra = 1 * pow(10, -5), int loop
= 1000)
{
    double arr[loop];
    arr[0] = x0;
    double min = arr[0];
    for(int i = 0 ; i < loop; i++)
    {
        arr[i + 1] = arr[i] - alpha * grad(arr[i]);
        if(min > arr[i + 1])
        {
            min = arr[i + 1];
        }
        if(abs(grad(i + 1)) < gra)
        {
            break;
        }
    }
    return min;
}

int main()
{
    cout << "gia tri nho nhat la: ";
    cout << myGradient_Discent(0.1, 0.5) << endl;
}
```

# **Trả lời**: Dán kết quả thực thi với điểm khởi $x = 0.50000$, tham số học học (learning rate) $\gamma = 0.1$, số bước lặp $N =$ *1000* và sai số $\varepsilon = 10^{-5}$:

```
d:\Class\ky1_2122\ToánUD\thiCK>cd "d:\Class\ky1_2122\ToánUD\thiCK\" && g++ bai4.cpp -o bai4 && "d:\Class\ky1_2122\ToánUD\thiCK\"bai4
gia tri nho nhat la: 0
```

***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.




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

**# Trả lời:** dán kết quả vào bên dưới:

- Lời giải:

Ma trận chuyển đổi trạng thái P của hệ:

a, Xác định Ma trận chuyển đổi trạng thái P của hệ.

Gọi $S_1$ là trạng thái 0 của hệ thống

$S_2$ là trạng thái 1 của hệ thống.

Ta có:

$$\begin{array}{c} \\ S_1 \\ S_2 \end{array} \begin{array}{c} \begin{array}{cc} S_1 & S_2 \end{array} \\ \begin{bmatrix} 0,25 & 0,65 \\ 0,75 & 0,35 \end{bmatrix} \end{array}$$

Vậy ma trận đó là: $\begin{bmatrix} 0,25 & 0,65 \\ 0,75 & 0,35 \end{bmatrix}$

b) (0.5) 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:

* Lời giải:

- Để xác suất là lớn nhất thì xu hướng vận hành của hệ thống là đi từ 0 sang 1 và ngược lại.

- Sau hai giai đoạn vận hành khi hệ thống bắt đầu ở trạng thái 0 là:

+ giai đoạn 1:

P1 = 0.75 * 0.65 = 0.4875

+ giai đoạn 2:

P2 = 0.75 * 0.65 = 0.4875

Vậy xác suất lớn nhất là:

P = P1 * P2 = 0.23765625

Đà Nẵng, ngày 01 tháng 12 năm 2021

**GIẢNG VIÊN BIÊN SOẠN ĐỀ THI**

**KHOA CÔNG NGHỆ THÔNG TIN**

**( đã duyệt)**