MỘT SỐ BÀI TOÁN VỀ QUY HOẠCH ĐỘNG

Bài 1: (`TONG.cpp`**)** Cho dãy số gồm N phần tử (0< N < 100): x_1, x_2, x_3, …, x_3 . Hãy tìm dãy con có tổng là s.

Dữ liệu vào:

- Dòng thứ 1: giá trị tổng s.
- Dòng thứ 2: số phần tử của dãy số.
- Dòng thứ 3: các phần tử của dãy số

Dữ liệu ra: Xuất ra các phần tử của mảng con.

TONG.INP	TONG.OUT
20 7 5 4 1 2 6 9 4	5 4 2 9

Ý tưởng :

- Ban đầu ta khởi tạo mảng d và mảng c. Trong đó mảng d dùng để đánh dấu các tổng có thể lập được. Và mảng c dùng để lưu vị trí phần tử của mảng tạo thành tổng đó.
- tổng j được lập nghĩa là d[j] = 1 nếu như trước đó d[j] = 0 tức tổng j chưa được thành lập (tránh trùng lặp ta gặp lại dạng trùng lặp ở bài 4) và d[j – a[i]] = 1 có nghĩa là đã tồn tại tổng j – a[i];
- Tổng s lập được khi tại vị trí j = s trong mảng d tức d[j] = 1;

i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
d	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- Duyệt qua từng phần tử của mảng. Xét xem những tổng nào có thể lập được từ phần tử tương ứng đó với những tổng đã lập được trước đó.

Ví dụ : Xét phần tử a[1] = 5 và a[2] = 4

- Cấu hình khởi tạo ban đầu d là d[0] = 1, tức tổng bằng 0 dĩ nhiên là d[0] = 1; có nghĩa là: 0 + a[i] thì tổng lập được là chính a[i].
- Đối với phần tử thứ nhất a[1] thì tổng lập được duy nhất là 5.
- Tiếp tục xét phần tử thứ 2 a[2] = 4. Là 9 vì cộng tổng ta lập trước là 5. Và tổng tiếp theo ta lập được là chính là d[0] + a[2]; có nghia là 0 + 4 = 4.

i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
d	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
c	0	0	0	2	1	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0

- Tiếp tục quá trình trên ta xem lập được bảng như sau:

i	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
d	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
c	3	4	4	2	1	3	4	4	2	3	4	4	5	5	5	5	5	5	6	6

- Truy vết (Ta tìm vị trí phần tử) ta làm như: chạy ngược từ i= 20

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
d	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
c	3	4	4	2	1	3	4	4	2	3	4	4	5	5	5	5	5	5	6	6

a[c[20]] = a[6] = 9

a[c[20 - 9]] = a[4] = 2

a[c[11 - 2]] = a[2] = 4

c[c[9 - 4]] = a[1] =5

Tuy nhiên ở bài này vị trí của các phần tử thay đổi theo thứ tự đảo ngược. Để in ra vị trí không thay đổi ta cần một mảng s[i] để đánh dấu vị trí mảng cần lấy khởi tạo ban đầu mảng s là 2, ta lấy ra phần tử nào thì đánh dấu là 1.

i	1	2	3	4	5	6	7
a[i]	5	4	1	2	6	9	4
s[i]	1	1	2	1	2	1	2

```
// day_con_tong_bang_s.cpp : Defines the entry point for the console application.
//
#include <bits/stdc++.h>

#define MAX 1000

using namespace std;

typedef int Mang[MAX];

int n, m, sum;
Mang a, d, c, s;

// Doc file
void ReadFile()
{
    ifstream f;
    f.open("TONG.INP");

    f >> sum;
    f >> n;

    for(int i = 1; i <= n; i++)
    {
        f >> a[i];
        // Mang danh dau phan tu lay ra
        s[i] = 2;
    }

    f.close();

}

// Khoi tao mang ban dau
void Create()
{
```

```
    for(int i = 0; i <= sum; i++)
    {
        d[i] = 0; c[i] = 0;
    }
}

// Tao mang quy hoach dong
void CreateSum()
{
    ReadFile();
    Create();

    d[0] = 1;
    for(int i = 1; i <= n; i++)
    {
        for(int j = sum; j >= a[i]; j--)
        { //Tổng trước đã hình thành d[j - a[i]] == 1 và d[i] == 0 để xem tổng này
đã lập chưa
            if(d[j] == 0 && d[j - a[i]] == 1)
            {
                d[j] = 1;
                c[j] = i;
            }
        }
    }
}


void Solve()
{

    CreateSum();

    // gáng m để tổng không thay đổi
    m = sum;

    // Đánh dấu phần tử lấy ra của mảng
    while(c[m] > 0)
    {
        s[c[m]] = 1;
        m = m - a[c[m]];
    }

    ofstream f;
    f.open("TONG.OUT");

    for(int i = 1; i <= n; i++)
    {
        if(s[i] == 1)
            f << a[i] << " ";
    }

    f.close();
}

int main(
{
```

```
    Solve();
    return 0;
}
```

Dựa trên ý tưởng của bài toán tìm dãy con có tổng bằng s ta xây dựng thuật toán

Bài 2: (CHIAKEO.cpp**)** Cho N gói kẹo (0< N < 100) có số viên kẹo lần lượt là: x_1, x_2, x_3, …, x_3 . Hãy tìm cách chia số kẹo trên thành hai phần mà chênh lệch giữa hai phần là nhỏ nhất

Dữ liệu vào:

- Dòng thứ 1: Số gói kẹo 5
- Dòng thứ 2: số viên kẹo của mỗi gói.

Dữ liệu ra:

- Dòng thứ 1: số kẹo chênh lệch
- Dòng thứ 2, 3: Dãy con tương ứng của mỗi phần

Ví dụ:

CHIAKEO.INP	CHIAKEO.OUT
5 13 5 7 4 11	0 13 7 5 4 11

```
#include <bits/stdc++.h>

#define MAX 1000

using namespace std;

typedef int Mang[MAX];

int n, sum, m, flag;
Mang a, d, c, s;

// Doc file
void ReadFile()
{
    sum = 0;
    ifstream f;
    f.open("CHIAKEO.INP");

    f >> n;

    for(int i = 1; i <= n; i++)
    {
        f >> a[i];
        sum += a[i];
        s[i] = 2;
    }

    f.close();

}
```

```
// Khoi tao mang ban dau
void Create()
{
    for(int i = 0; i <= sum; i++)
    {
        d[i] = 0; c[i] = 0;
    }

}

// Tao mang quy hoach dong
void CreateSum()
{
    ReadFile();
    Create();

    d[0] = 1;
    for(int i = 1; i <= n; i++)
    {
        for(int j = sum; j >= a[i]; j--)
        {
            if(d[j] == 0 && d[j - a[i]] == 1)
            {
                d[j] = 1;
                c[j] = i;
            }
        }
    }
}

void Solve()
{

    CreateSum();

    m = sum/2;

    while(d[m] == 0)
    {
        m = m - 1;
    }

    //danh dau phan tu cua mang con
    while(c[m] > 0)
    {
        s[c[m]] = 1;
        m = m - a[c[m]];
    }

    ofstream f;
    f.open("CHIAKEO.OUT");

    f << sum/2 << endl;
    f << sum - sum/2 << endl;

    // lay ra phan thu 1
    for(int i = 1; i <= n; i++)
```

```
    {
        if(s[i] == 1) f << a[i] << " ";
    }

    f << endl;

    // lay ra phan thu 2
    for(int i = 1; i <= n; i++)
    {
        if(s[i] == 2) f << a[i] << " ";
    }


    f.close();
}
```

Bài 3: Có một người đi mua hàng, anh ta có N đồng tiền d1 , d2 ,.., dN . Người bán hàng có M đồng tiền b1 , b2 ,.., Anh ta muốn mua một mặt hàng với giá trị W. Hỏi cuộc mua bán có thể diễn ra được không ?. Nếu W mà vượt quá tổng số tiền của người mua hoặc người bán không có tiền để trả lại coi như giao dịch thất bại

Giới hạn: M, N <=100 và di, bj <=100.

Dữ liệu vào:

- Dòng thứ nhất chứa 3 số nguyên W, N, M
- Dòng số 2 là các tờ tiền của người thứ mua.
- Dòng số 3 là các tờ tiền của người bán.

Dữ liệu đầu ra:

- Dòng thứ nhất: thông báo YES hoặc NO (Giao dịch thành công hay thất bại).
- Dòng thứ hai: số tiền phải thối của người bán

BUY.INP	BUY.OUT
50 5 9 5 7 10 9 15 6 5 7 12 6 20 10 15 9 2	YES 2

Bài 4: **(**`MONEY.cpp`**)** Một người đi lấy tiền ở một ngân hàng. Anh ta cần lấy một khoản đúng M đồng. Ngân hàng có N đồng:

tiền A 1 ,A 2 ,.., A N . Hỏi ngân hàng có bao nhiêu cách trả tiền mỗi loại chỉ lấy nhiều nhất 1 đồng.

Dữ liệu vào trong file: "MONEY.INP"; có dạng:

+ Dòng đầu là hai số N và M (N <=100, M <= 10000)
+ Các dòng tiếp theo là các phần tử của mảng A.

Kết quả ra file: "MONEY.OUT" gồm một dòng duy nhất là số cách trả tiền (Số cách trả tiền < Maxlongint).

MONEY.OUT	MONEY.INP
5 10 1 2 3 4 5	3

Bài toán được xây dựng trên cơ sở đếm số lần lập được một tổng từ các phần tử của mảng. Nếu d[s] = 0 tức không lập được tổng, ngược lại tổng ta lập được sẽ là d[j] = d[j] + d[j – a[i]];

```
#include <bits/stdc++.h>

#define MAX 1000

using namespace std;

typedef int Mang[MAX];

Mang a, d, c;
int n, s;

void ReadFile()
{
    ifstream f;
    f.open("MONEY.INP");

    f >> n >> s;

    for(int i = 1; i <= n; i++)
    {
        f >> a[i];
    }

    f.close();
}

void Solve()
{
    ReadFile();


    for(int i = 0; i <= s; i++) d[i] = 0;

    d[0] = 1;

      // tính xem mỗi tổng xuất hiện mấy lần
    for(int i = 1; i <=n; i++)
    {
        for(int j = s; j >= a[i]; j--)
        {
            // số cách tính của tổng j là bằng số cách tích của tổng j – a[i] và
            số cách tính tổng j trước đó. (Phép cộng trong xác suất).
            d[j] = d[j] + d[j - a[i]];
        }
    }

    ofstream f;
    f.open("MONEY.OUT");

    f << d[s]; // d[s] = 0 thi không thể lập được tổng

    f.close();
}
```

Bài 5: (PHANTHUONG.cpp**)** Cho dãy số gồm N phần tử (0< N < 100): x_1, x_2, x_3, …, x_3. Hãy chon những phần tử sao cho tổng các phần tử đã chon là lớn nhất. Nguyên tắc chọn là 3 phần tử được chọn không thể liên tiếp nhau.

Dữ liệu vào: Các phần tử của dãy số.

Dữ liệu ra:

- Dòng 1 : Tổng của dãy con.
- Dòng 2: Các phần tử của dãy con.

PHANTHUONG.INP	PHANTHUONG.OUT
6 5 6 9 1 3 5	23 6 9 3 5

```
#include <bits/stdc++.h>

using namespace std;

#define MAX 1000

typedef int Mang[MAX];

int n, u, v, t;
Mang x, c, s, flag;

void ReadFile()
{
    ifstream f;
    f.open("PHANTHUONG.INP");

    f >> n;

    if(n == 1)
    {
        f >> u;
        f.close();
        return;
    }

    if(n == 2)
    {
        f >> u;
        f >> v;
        f.close();
        return;
    }

    for(int i = 1; i <= n; i++)
    {
        f >> x[i];
    }
    f.close();
```

```
}

// danh dấu các phần quà sẽ lấy
// s[3] dùng để lưu giá trị của tổng các phần quà
void CreateSum()
{
    // khoi tao mang danh dau
    for(int i = 1; i <= n; i++) c[i] = 0;

    c[1] = -1;
    c[2] = -1;
    s[0] = 0;
    s[1] = x[1];
    s[2] = x[1] + x[2];

    for(int i = 3; i <= n; i++)
    {
        s[3] = s[2]; c[i] = i - 1;
        if(s[3] < x[i] + s[1])
        {
            s[3] = x[i] + s[1];
            c[i] = i - 2;
        }

        if(s[3] <  x[i] + x[2] + s[0])
        {
            s[3] = x[i] + x[2] + s[0];
            c[i] = i - 3;
        }

        x[1] = x[2];
        x[2] = x[i];
        s[0] = s[1];
        s[1] = s[2];
        s[2] = s[3];
    }

}

void Trace()
{
    t = 0;
    int a = n;

    lap:
        int b = c[a];
        for(int i = a; i >= b + 2; i--)
        {
            t = t + 1;
            flag[t] = i;
        }
        a = b;

    if(a > 0) goto lap;
}

void Solve()
```

```
{
    ReadFile();

    ofstream f;
    f.open("PHANTHUONG.OUT");

    // trường hợp chỉ có một phần quà
    if(n == 1)
    {
        f << u;
        return;
    }
    // trường hợp 2 phần quà
    if(n == 2)
    {
        f << u + v << endl;
        f << u << " " << v;
        f.close();
        return;
    }

    // trường hợp nhiều hơn 2 phần quà
    CreateSum();
    Trace();

    f << s[3] << " " << n << endl;

    int dem = 0;
    for(int i = t; i >= 1; i--)
    {
        dem++;
        f << flag[i] << " ";
        if(dem % 10 == 0) f << endl;
    }

    f.close();

}

int main()
{

    Solve();
    return 0;
}
```

Bài 6: (CHIAKEO.cpp) Có n gói kẹo (n là số nguyên dương), gói kẹo thứ i có x_i viên kẹo.
Yêu cầu: Hãy viết chương trình chia số kẹo trên thành nhiều phần nhất sao cho tổng số viên kẹo trong các phần là bằng nhau.
Dữ liệu vào: "CHIAKEO.INP"
- Dòng đầu ghi n (n > 1000).
- Dòng tiếp theo ghi n số nguyên dương x_i, x_2, …., x_n

Dữ liệu ra: "CHIAKEO.OUT"

- Ghi số nguyên dương là số phần chia được theo đề bài.
- Nếu không chia được ghi là 1.

CHIAKEO.INP	CHIAKEO.OUT
8 10 2 6 2 5 2 1 2	3
4 2 5 4 1	2
3 7 3 2	1

```
#include <bits/stdc++.h>

using namespace std;

#define MAX 100

typedef int Mang[MAX];

int n, Max = -1, sum, dem = 0;
Mang x, d;

void ReadFile()
{
    sum = 0;

    ifstream f;
    f.open("CHIAKEO.INP");
    f >> n;

    for(int i = 1; i <= n; i++)
    {
        f >> x[i];
        sum += x[i];
        if(Max < x[i]) Max = x[i];
    }

    f.close();

}

void Sort()
{
    for(int i = 1; i < n; i++)
    {
        for(int j = i + 1; j <= n; j++)
        {
            if(x[i] < x[j])
            {
                int tam = x[i];
                x[i] = x[j];
                x[j] = tam;
```

```
            }
        }
    }
}

void Solve()
{
    ofstream f;
    f.open("CHIAKEO.OUT");

    ReadFile();

    for(int i = 0; i <= sum; i++)  d[i] = 0;

    d[0] = 1;

    for(int i = 1; i <= n; i++)
    {
        for(int j = sum; j >= x[i]; j--)
        {
            if(d[j] == 0 && d[j - x[i]] == 1)
            {
                d[j] = 1;
            }
        }
    }

    for(int t = Max; t <= sum; t++)
    {
        if(sum % t == 0)
        {
            int nguyen = sum/t;
            f << nguyen;
            f.close();
            return;
        }
    }
}
```

Bài 7: (THANHGO.cpp**)** Trong một buổi cắm trại của lớp, bạn An mua N thanh gỗ có độ dài mỗi thanh là L. Khi cắm trại, các bạn của An cưa các thanh gỗ ra một cách ngẫu nhiên (có độ dài là số nguyên).

Về sau các bạn có ý định gắn các mẩu con để khôi phục lại các thanh gỗ ban đầu nhưng lại quên mất độ dài L. Họ đã quyết định nối lại các thanh gỗ sao cho chúng có độ dài bằng nhau.

Hãy giúp họ chọn cách nối sao cho chúng có độ dài như nhau và càng ngắn càng tốt.

Dữ liệu vào: cho trong file văn bản **THANHGO.INP**:

- Dòng đầu ghi số N (N≤50) là số lượng các mẩu gỗ.
- N dòng tiếp theo mỗi dòng ghi số nguyên L_i ($1 \le L_i \le 100$, $1 \le i \le N$) thể hiện độ dài của mẩu gỗ thứ i.

Kết quả: Ghi ra file văn bản **THANHGO.OUT**

- Dòng đầu tiên ghi độ dài ngắn nhất tìm được.
- Trên mỗi dòng ghi số hiệu các mẩu gỗ dùng để ghép thành thanh gỗ đó.

Ví dụ:

THANHGO.INP	THANHGO.OUT
10 2 3 5 2 7 4 6 1 3 3	9 3 6 7 2 5 4 9 10 1 8

```
#include<bits/stdc++.h>

using namespace std;

#define MAX 10000

typedef int Mang[MAX];

/*
    x la mang input
    d danh dau tong
    dd danh dau phan tu lap tong
    s danh dau day con
    tempt luu vi tri
    c luu vi tri danh con
*/

Mang x, d, dd, s, tempt, c;

int n, dem, sum;
```

```
// doc file
void ReadFile()
{
    sum = 0;

    ifstream f;
    f.open("THANHGO.INP");
    f >> n;

    for(int i = 1; i <= n; i++)
    {
        f >> x[i];
        tempt[i] = i;
        sum += x[i];
    }

    f.close();
}

void Print(int k, Mang a)
{
    cout << "\n\n\t\t ==============XUAT MANG================\n\n";
    for(int i = 1; i <= k; i++) cout << a[i] << " ";
    cout << endl;
}

// Khoi tao mang ban dau
void CreateEmpty(int t)
{
    for(int i = 0; i <= t; i++)
    {
         d[i] = 0;
         c[i] = 0;
    }
    d[0] = 1;
}

// Hoan doi vi tri cua a va b
void Swap(int &a, int &b)
{
    int tam = a;
    a = b;
    b = tam;
}

// Sap xep mang
void Sort()
{
    for(int i = 1; i < n; i++)
    {
        for(int j = i + 1; j <= n; j++)
        {
            if(x[i] < x[j])
            {
                Swap(x[i], x[j]);
                Swap(tempt[i], tempt[j]);
            }
```

```
        }
    }
}

bool Check(int t)
{
    CreateEmpty(t);

    for(int i = 1; i <= n; i++)
    {
        if(dd[i] == 0) // neu phan tu thu i chua chua lap thi moi lap tong
        {
            for(int j = t; j >= x[i]; j--)
            {
                if(d[j] == 0 && d[j - x[i]] == 1)
                {
                    d[j] = 1;
                    c[j] = i;
                }
            }
        }
    }

    // Truy vet
    if(d[t] == 1) // tong t duoc thanh lap
    {
        int k = t;
        dem++;
        while(c[k] != 0)
        {
            s[c[k]] = dem;
            dd[c[k]] = 1;
            k = k - x[c[k]];
        }
        return true;
    }
    return false;

}

void CreateSum()
{
    int t = x[1];
    for(int i = t; i <= sum; i++)
    {
        if(sum % i == 0)
        {
            // khoi tao mang danh daau phan tu cua
            // tung mang con va mang danh dau phan tu da lay
            for(int j = 0; j <= i; j++)
            {
                dd[j] = 0;
                s[j] = 0;
            }

            dem = 0; // gan bien dem bang 0
            bool kt = Check(i);
```

```
            int k = sum/i;

            while(kt == true && k != dem)
            {
                kt = Check(i);
            }

            if(dem == k)
            {
                ofstream  f;
                f.open("THANHGO.OUT");

                f << i << endl;

                for(int j = 1; j <= k; j++)
                {
                    for(int l = 1; l <= n; l++)
                    {
                        if(s[l] == j) f << tempt[l] << " ";
                    }
                    f << endl;
                }
                f.close();
                return;
            }

        }
    }
}

int main()
{
    ReadFile();
    Sort();
    Optimize();

    Print(n, x);
    return 0;
}
```

Bài 8: **(**PHATGAO.cpp**)** Một kho chứa gạo cần dùng đủ để phát cho các hộ gia đình bị thiên tai ở một làng nọ. Trong kho chứa N loại bao gạo với khối lượng khác nhau K[1], K[2],.., K[N] (0<K[i]≤100). Mỗi người đại diện cho một hộ gia đình có M người được phát một số lượng gạo tính theo đầu người là T kg/người (T mod 5 = 0). Bạn hãy viết chương trình giúp cho thủ kho có thể dễ dàng biết được phải phát bao nhiêu bao gạo, gồm những loại nào cho mỗi hộ gia đình một cách nhanh chóng. *Lưu ý mỗi lần phát một bao gạo phải phát nguyên bao chứ không chia nhỏ bao.*

Dữ liệu vào: cho file **PHATGAO.INP** gồm các dòng:

- Dòng đầu tiên chứa 3 số nguyên N, M, T (0<N,M,T ≤ 100), mỗi số cách nhau ít nhất một khoảng cách.
- Dòng hai chứa N số nguyên K[1], K[2],.., K[N]

Dữ liệu ra: ghi ra file **PHATGAO.OUT** gồm một trong hai yếu tố sau:

- Dòng một chứa số lượng bao gạo cần phát.

- Dòng hai chứa N số nguyên không âm ứng với khối lượng mỗi bao gạo được phát.

Ví dụ:

PHATGAO.INP	PHATGAO.OUT
5 3 20 5 10 15 20 30	5 1 2 1 1 0

Hay

PHATGAO.INP	PHATGAO.OUT
5 3 20 5 10 15 20 30	3 0 1 0 1 1

```
#include <bits/stdc++.h>

using namespace std;

#define MAX 100

typedef int Mang[MAX];

// khai bao bien
int M, N, T, x, dem;
Mang K, d, c, s, vt;

// doc file
void ReadFile()
{
    ifstream f;
    f.open("PHATGAO.INP");

    f >> N;
    f >> M;
    f >> T;

    for(int i = 1; i <= N; i++)
    {
        f >> K[i];
        vt[i] = i;
        s[i] = 0;
    }

    f.close();
}

void Swap(int &a, int &b)
{
```

```
    int tam = a;
    a = b;
    b = tam;
}

void Sort()
{
    for(int i = 1; i < N; i++)
    {
        for(int j = i + 1; j <= N; j++)
        {
            if(K[i] < K[j])
            {
                Swap(K[i], K[j]);
                Swap(vt[i], vt[j]);
            }
        }
    }
}

// khoi tao rong mang danh dau d, c;
void CreateEmpty()
{
    for(int i = 0; i <= x; i++)
    {
        d[i] = 0;
        c[i] = 0;
    }
}

// truy vet
int Trace(int sum, int& dem)
{
    int m = sum;
    while(d[m] == 0) m--;

    int n = x - m;

    while(c[m] > 0)
    {
        dem++;
        s[c[m]] = s[c[m]] + 1;
        m = m - K[c[m]];
    }

    return n;
}

// Toi uu hoa
void Optimize()
{
    CreateEmpty(); d[0] = 1;
    Sort();

    x = M * T;
    // lap tat ca cac tong tu 1 den M * T
    for(int i = 1; i <= N; i++)
```

```
    {
        for(int j = x; j >= K[i]; j--)
        {
            if(d[j] == 0 && d[j - K[i]] == 1)
            {
                d[j] = 1;
                c[j] = i;
            }
        }
    }

    // goi lai ham truy vet
    dem = 0;
    int n = Trace(x, dem);

    while(n != 0)
    {
        n = Trace(n, dem);
    }
}

void Solve()
{
    ReadFile();
    cout << "\n\t\t=============== XUAT MANG ===============\n\n";
    for(int i = 1; i <= N; i++) cout << K[i] << " ";
    cout << endl;

    Optimize();

    ofstream f;
    f.open("PHATGAO.OUT");

    f << dem << endl;
    for(int i = 1; i <= N; i++) f << s[vt[i]] << " ";

    f.close();
}

int main()
{
    Solve();
    return 0;
}
```

Bài 9: (INCSEQ.cpp**)**Cho dãy số A gồm N phần tử (0< N < 100): x_1, x_2, x_3, …, x_3. Một dãy con đơn điệu tăng của A là dãy gồm các phần tử tăng dần nhưng vẫn dữ nguyên trật tự.
Yêu cầu: Tìm dãy con tăng đơn điệu của A có số phần tử lớn nhất.
Dữ liệu vào:
- Dòng đầu tiên : số nguyên N
- Dòng thứ hai: dãy số nguyên A
Dữ liệu ra: Dãy con tìm được

INCSEQ.INP	INCSEQ.OUT

1 2 10 4 5 2 6 7 3 4 6 8	7 1 2 4 5 6 7 8

```
// day_con_tang.cpp : Defines the entry point for the console application.
//

#include <bits/stdc++.h>

using namespace std;

#define MAX 100

typedef int Mang[MAX];
Mang x, c, L;
int n, Max = 1, vt = -1;

void ReadFile()
{
    ifstream f;
    f.open("DAYTANG.INP");

    int  i = 0;
    while(!f.eof())
    {
        f >> x[i];
        c[i] = 0;
        i++;
    }

    n = i - 1;
}

void Print(Mang a)
{
    for(int i = 0; i < n; i++)
    {
        cout << a[i] <<  " ";
    }
    cout << endl;
}

void Optimize()
{
    L[0] = 1;
    for(int i = 1; i < n; i++)
    {
        int lmax = 0;
        for(int j = 0; j < i; j++)
        {

            if(x[i] > x[j])
            {
                if(lmax < L[j])
                {
```

```
                    lmax = L[j];
                }
            }
        }

        L[i] = lmax + 1;
        if(Max < L[i])
        {
            Max = L[i];
            vt = i;
        }
    }

}

void Solve()
{
    ReadFile();
    Optimize();

    ofstream f;
    f.open("DAYTANG.OUT");

    f << L[vt] << endl;

    int k = vt;

    while(k >= 0)
    {
        if(L[k] == Max)
        {
            c[k] = 1;
            Max--;
        }
        k--;
    }

    for(int i = 0; i < n; i++)
    {
        if(c[i] != 0) f << x[i] << " ";
    }

    f.close();
}


int main()
{
    Solve();
    return 0;
}
```

<u>Câu 10:</u> Cho dãy số A gồm N phần tử (0< N < 100): x_1, x_2, x_3, …, x_3. Một dãy con đơn điệu tăng của A là dãy gồm các phần tử tăng dần nhưng vẫn dữ nguyên trật tự.

Yêu cầu: Tìm dãy con tăng đơn điệu của A có số phần tử lớn nhất bằng cách chỉ xóa tối đa một phần tử
Dữ liệu vào:
- Dòng đầu tiên : số nguyên N
- Dòng thứ hai: dãy số nguyên A
Dữ liệu ra: Dãy con tìm được

```
// day_loi.cpp : Defines the entry point for the console application.
//
#include "stdio.h"
#include "conio.h"

#define INPUT "INCSEQ.INP"
#define OUTPUT "INCSEQ.OUT"
#define MAX 100

typedef int Mang[MAX];

typedef int Mang2C[MAX][MAX];

void ReadFile(int &n, Mang &x)
{
	FILE * fr;
	errno_t err = fopen_s(&fr, INPUT, "r");

	if(err == 0)
	{
		fscanf_s(fr, "%d", &n);
		for(int i = 0; i < n; i++)
		{
			fscanf_s(fr, "%d", &x[i]);
		}
		fclose(fr);
	}
	else
	{
		printf_s("Error, Can't open file ... !");
	}
}

void  PrintItems(int n, Mang x)
{
	printf_s("Items: ");
	for(int i = 0; i < n; i++)
	{
		printf_s("%3d", x[i]);
	}
}

void Solve(int n, Mang x, Mang &s)
{
	int j = 0;
	int dem = 0, sum = 1;

	int tam = x[0];
```

```
	s[0] = sum;

	for(int i = 1; i < n; i++)
	{
		if(dem <= 1)
		{
			if(x[i] > tam)
			{
				tam = x[i];
				sum++;
			}
			else
			{
				dem++;
				if(x[i + 1] < tam)
				{
					dem++;
					tam = x[i];
				}
			}
		}

		if(dem  > 1)
		{
			sum = 1;
			dem = 0;
			tam = x[i];
		}

		s[i] = sum;
	}


}

void WriteFile(int n, Mang s, Mang x)
{

	int b = 0, max = s[0];

	for(int i = 1; i < n; i++)
	{
		if(s[i] > max)
		{
			max = s[i];
			b = i;
		}

	}

	int j = 0; Mang c;

	FILE * fw;

	fopen_s(&fw, OUTPUT, "w");

	int k = b;
```

```
	while(s[k] != 1)
	{
		if(x[k] > x[k - 1])
		{
			c[j] = x[k];
			j++;
		}
		k--;
	}

	c[j] = x[k];

	for(int t = j; t >= 0; t--)
	{
		fprintf_s(fw, "%3d", c[t]);
	}

	fclose(fw);

}

int main()
{
	Mang x, s;
	int n;

	ReadFile(n, x);
	PrintItems(n, x);

	printf_s("\n");
	Solve(n, x, s);
	PrintItems(n, s);

	printf_s("\n");
	WriteFile(n, s, x);


	_getch();
	return 0;
}
```

Câu 11: **(**DUONGDI.cpp**)** Sân chơi là một mặt phẳng chia ra thành N hàng đánh số từ 1 đến N (1<N<100). Ở hàng thứ i (1≤ i ≤ N) có i ô điểm có giá trị cho trước là những số nguyên dương (không vượt quá 1000). Trò chơi là chọn một lộ trình với ô xuất phát là ô ở hàng thứ nhất, lần lượt đi qua một trong 2 ô lân cận ở hàng tiếp theo (theo hướng mũi tên) cho đến khi đến được một ô ở hàng cuối cùng và thu nhặt các điểm số có ở các ô trên đường đi qua (lộ trình sẽ thăm đúng N ô) *(Hình vẽ dưới minh họa cho một ví dụ với N=4).*

Cho trước một bảng biểu thị giá trị điểm số các ô trên từng hàng. Hãy lập trình tìm một lộ trình hợp quy định của luật chơi và thu được điểm số cao nhất.

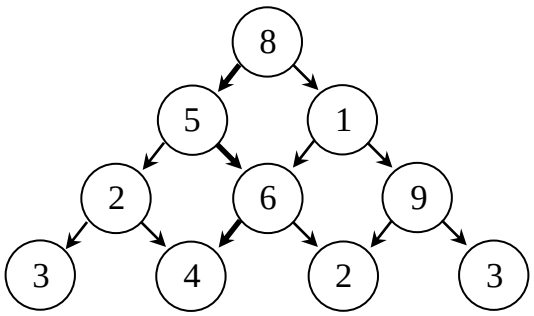
Dữ liệu vào là tệp DUONGDI.INP có cấu trúc như sau:

- Dòng thứ nhất chứa số tự nhiên N;

- N dòng tiếp theo sẽ chứa các giá trị điểm số trên các ô điểm ở dòng tương ứng. Dòng thứ i sẽ có i giá trị. Các giá trị cách nhau một khoảng trắng.

Dữ liệu ra là tệp DUONGDI.OUT gồm 2 dòng:

- Dòng thứ nhất chứa giá trị tổng điểm lớn nhất thu được theo lộ trình tối ưu;
- Dòng thứ 2 chứa N số nguyên là giá trị các ô điểm mà lộ trình tối ưu đi qua.



Ví dụ:

DUONGDI.INP
4 8 5 1 2 6 9 3 4 2 3

DUONGDI.OUT
23 8 5 6 4

```
// duong_di.cpp : Defines the entry point for the console application.
//
#include "conio.h"
#include "stdio.h"

#define MAX 100
#define INPUT "DUONGDI_INP.txt"
#define OUTPUT "DUONGDI_OUT.txt"

typedef int Matrix[MAX][MAX];
typedef int Mang[MAX];

// Doc file
void ReadFile(int &n, Matrix &a)
{
	FILE * fr;

	errno_t err = fopen_s(&fr, INPUT, "r");

	if(err == 0)
	{
		fscanf_s(fr, "%d", &n);
```

```
			for(int i = 0; i < n; i++)
			{
				for(int j = 0; j <= i; j++)
				{
					fscanf_s(fr, "%d", &a[i][j]);
				}
			}
	}
}

// Tim phan tu lon nhat
int MaxItem(int a, int b)
{
	return a > b ? a : b;
}

//Quy hoach dong
void Optimize(int n, Matrix a, Matrix &L)
{

	L[0][0] = a[0][0];

	for(int i = 1; i < n; i++)
	{
		for(int j = 0; j <= i; j++)
		{
			if(j == 0) L[i][j] = a[i][j] + L[i - 1][j];
			else if(i == j) L[i][j] = a[i][j] + L[i - 1][j - 1];
			else L[i][j] = MaxItem(L[i - 1][j - 1], L[i - 1][j]) + a[i][j];
		}
	}
	for(int i = 0; i < n; i++)
	{
		for(int j = 0; j <= i; j++)
		{
			printf_s("%3d", L[i][j]);
		}
		printf_s("\n");
	}

}

// tim phuong an cua bai toan
void Solve(int n, Matrix a, Matrix L)
{

	int x, y, k = 0;

	FILE * fw;

	fopen_s(&fw, OUTPUT, "w");

	int tam = a[n - 1][0];

	for(int k = 1; k < n; k++)
	{
		if(tam < L[n - 1][k])
```

```
			{
				tam = L[n - 1][k];
				y = k;
			}
		}

		Mang c;
		x = n - 1;
		int j = 0;

		c[j] = a[x][y];

		fprintf_s(fw, "%d \n", L[x][y]);

		for(int i = x - 1; i >= 1; i--)
		{
			j++;
			if(L[i][y] > L[i][y - 1])
			{
				c[j] = a[i][y];
			}
			else
			{
				c[j] = a[i][y - 1];
				y = y - 1;
			}
		}

		j++;
		c[j] = a[0][0];

		fprintf_s(fw, "%d ", c[j]);

		for(int t = j - 1; t >= 0; t--)
		{
			fprintf_s(fw, "----> %d ", c[t]);
		}

		fclose(fw);
}

// Ham main
int main()
{
		Matrix a, L;
		int n;

		ReadFile(n, a);
		Process(n, a, L);
		Solve(n, a, L);

		_getch();
		return 0;
}
```

Câu 12: ROBOT (ROBOT.cpp**)**Cho một lưới ô vuông cấp NxN (1≤ N≤ 100). Trên mỗi ô của lưới người ta ghi một số nguyên a (1≤ a ≤ 100) được gọi là số điểm của ô đó. Một chú robot đang ở ô

(i,j) của lưới chỉ được di chuyển đến một trong hai ô (i+1,j) hoặc ô (i,j+1). Robot đi đến ô nào thì nhận được số điểm tương ứng ở ô đó. Hãy tìm cho Robot một đường đi từ ô (1,1) đến ô (N,N) theo nguyên tắc trên và tích lũy được số điểm nhiều nhất?

Dữ liệu vào: Từ file văn bản ROBOT.INP,

Dòng đầu tiên ghi 1 số nguyên dương N.

N dòng tiếp theo mỗi dòng ghi N số nguyên.

Kết quả: Ghi ra file văn bản ROBOT.OUT, dòng đầu tiên ghi một số S là số điểm mà robot tích lũy được. Từ dòng thứ 2 trở đi mỗi dòng ghi 2 số nguyên dương là toạ độ các ô theo thứ tự trên đường đi của robot (Nếu có nhiều phương án đúng chỉ cần ghi ra một phương án đúng).

Ví dụ:

ROBOT.INP	ROBOT.OUT
4 5 1 3 4 6 7 9 1 1 1 8 4 1 3 4 3	42 1 1 2 1 2 2 2 3 3 3 4 3 4 4

```
// ro_bot.cpp : Defines the entry point for the console application.
//
#include "stdio.h"
#include "conio.h"

#define MAX 100
#define INPUT "ROBOT_INP.txt"
#define OUTPUT "ROBOT_OUT.txt"

typedef int Matrix[MAX][MAX]; // Khai bao mang 2 chieu

struct Point // Cau truc diem
{
	int x;
	int y;
};

typedef Point ListPoint[MAX]; // Mang luu cac diem di qua

// Doc file
void ReadFile(int &n, Matrix &a)
{
	FILE * fr;

	errno_t err = fopen_s(&fr, INPUT, "r");

	if(err == 0)
	{
		fscanf_s(fr, "%d", &n);
```

```
		for(int i = 0; i < n; i++)
		{
			for(int j = 0; j < n; j++)
			{
				fscanf_s(fr, "%d", &a[i][j]);
			}
		}
	}
	else
	{
		printf_s("Error, Can't open file...!");
	}
}

// Tim phan tu max
int MaxItem(int a, int b)
{
	return a > b ? a : b;
}

//Xuat ra ma tran
void PrintMatrix(int n, Matrix a)
{
	printf_s("MATRIX: \n");
	for(int i = 0; i < n; i++)
	{
		for(int j = 0; j < n; j++)
		{
			printf_s("%3d", a[i][j]);
		}
		printf_s("\n");
	}
}

// Tao bang quy hoach dong
void Process(int n, Matrix a, Matrix &L)
{
	L[0][0] = a[0][0];

	for(int i = 1; i < n; i++)
	{
		L[0][i] = L[0][i - 1] + a[0][i];
		L[i][0] = L[i - 1][0] + a[i][0];
	}

	for(int i = 1; i < n; i++)
	{
		for(int j = 1; j < n; j++)
		{
			L[i][j] = MaxItem(L[i - 1][j], L[i][j - 1]) + a[i][j];
		}
	}

}

// Giai bai toan
void Solve(int n, Matrix L)
```

```
{
	ListPoint Points;
	int x1 = n - 1, y1 = n - 1;

	Points[0].x = x1 + 1; Points[0].y = y1 + 1;

	int i = 1;

	FILE * fw;
	fopen_s(&fw, OUTPUT, "w");

	fprintf_s(fw, "%3d\n", L[x1][y1]);

	while(x1 != 0 && y1 != 0)
	{
		if(L[x1 - 1][y1] > L[x1][y1 - 1])
		{
			Points[i].x = x1;
			Points[i].y = y1 + 1;
			x1--;
		}
		else
		{
			Points[i].x = x1 + 1;
			Points[i].y = y1;
			y1--;
		}
		i++;
	}

	Points[i].x = 1;
	Points[i].y = 1;

	for(int j = i; j >= 0; j--)
	{
		fprintf_s(fw, "%3d %3d\n", Points[j].x, Points[j].y);
	}
	fclose(fw);

}

int main()
{
	Matrix a, L;
	int n;

	ReadFile(n, a);
	Process(n, a, L);
	PrintMatrix(n, L);

	Solve(n, L);
	_getch();
	return 0;
}
```

Bài 12: (BOM.cpp) Địa hình khu vực khủng bố đóng quân có dạng ma trận x: n x n, x[i][j] = 1 là vị trí xe tăng cần phá hủy, x[i][j] = 0 là vị trí không có xe tăng đậu. 1 quả bom ở vị trí (i, j) khi nổ sẽ

phá hủy các vị trí (i, j), (i + 1, j), (i, j + 1), (i + 1, j + 1). Nếu ở vị trí đó có xe tăng nó cũng phá hủy theo. Hãy viết chương trình đặt bom sao cho số bom là nhỏ nhất.

BOM.INP	BOM.OUT
4 1 1 0 1 1 0 1 1 0 0 0 1 0 1 1 1	4 qua bom (1, 1) (1, 3) (3, 3) (4, 2)

```
#include <bits/stdc++.h>

#define MAX 100

using namespace std;

typedef int Mang[MAX][MAX];

int n;
Mang x, L;

int ReadFile()
{
    int s = 0;

    ifstream f;
    f.open("BOM.INP");

    f >> n;

    for(int i = 0; i <= n + 1; i++)
    {
        for(int j = 0; j <= n + 1; j++)
        {
            if(i == 0 ||  j == 0 || i == n + 1 || j == n + 1) x[i][j] = 0;
            else f >> x[i][j];
            if(x[i][j] == 1) s += x[i][j];
        }
    }

    f.close();

    return s;

}

void Create()
{
    for(int i = 0; i <= n + 1; i++)
    {
        for(int j = 0; j <= n + 1; j++)
```

```
        {
            L[i][j] = x[i][j] + x[i + 1][j] + x[i][j + 1] + x[i + 1][j + 1];
        }
    }
}

void Print(Mang a)
{
    printf("\n\n\t\t=================MATRIX=================\n\n");
    for(int i = 1; i <= n; i++)
    {
        for(int j = 1; j <= n; j++)
        {
            printf("%3d", a[i][j]);
        }
        cout << endl;
    }
}

void Optimize()
{

    Create();
    for(int i = 1; i <= n; i++)
    {
        for(int j = 1; j <= n; j++)
        {
            if(x[i][j] == 0 && (L[i][j] < L[i][j + 1] || L[i][j] < L[i + 1][j + 1]
|| L[i][j] < L[i + 1][j]))
            {
                L[i][j] = 0;
            }

            if(L[i][j] >= L[i][j + 1] && L[i][j] >= L[i + 1][j + 1] && L[i][j] >=
L[i + 1][j])
            {
                L[i][j + 1] = 0;
                L[i + 1][j + 1] = 0;
                L[i + 1][j] = 0;
            }

            if(x[i - 1][j - 1] != 0 && L[i - 1][j - 1] != 0) L[i][j] = 0;

        }
    }
}

void Solve()
{
    ReadFile();
    Print(x);
    Optimize();

    ofstream f;
    f.open("BOM.OUT");

    int dem = 0;
```

```
    for(int i = 1; i <= n; i++)
    {
        for(int j = 1; j <= n; j++)
        {
            if(L[i][j] != 0) dem++;
        }
    }

    f << dem << " qua bom" << endl;

    for(int i = 1; i <= n; i++)
    {
        for(int j = 1; j <= n; j++)
        {
            if(L[i][j] != 0) f << "(" << i << ", " << j << ") ";
        }
    }

    f.close();
}

int main()
{
    Solve();
    return 0;
}
```

Bài 13: (XAUCON.cpp) Tìm xâu con chung dài nhất. Xâu con chung được định nghĩa như sau: Nếu xóa đi một số kí tự của hai xâu thì hai xâu con còn lại của chúng bằng nhau.

XAUCON.INP	XAUCON.OUT
CEACEEC AECECA	ECEC

Hoặc

XAUCON.INP	XAUCON.OUT
CEACEEC AECECA	AEEC

```
// TIM_XAU_CON_MAX.cpp : Defines the entry point for the console application.
//

#include <bits/stdc++.h>

#define INPUT "XAU_INP.txt"
#define OUTPUT "XAU_OUT.txt"

#define SIZE 100
```

```
typedef char chuoi[SIZE];
typedef int mang[SIZE][SIZE];

// Doc file
void ReadFile(chuoi &a, chuoi &b)
{
	FILE * fr;

	errno_t error = fopen_s(&fr, INPUT, "r");

	if (error == 0)
	{
		fgets(a, SIZE, fr);
		fgets(b, SIZE, fr);
		fclose(fr);
	}
	else
	{
		printf_s("Error, Can't open file...!");
	}

}

// tim max
int Max(int a, int b)
{
	return a > b ? a : b;
}


void CreateTable(mang L, int m, int n)
{
	for (int i = 0; i <= m; i++)
	{
		for (int j = 0; j <= n; j++)
		{
			L[i][j] = 0;
		}
	}
}

void PrintItems(mang L, int m, int n)
{
	printf_s("\n\t\t\tTABLE: \n\n");
	for (int i = 0; i <= m; i++)
	{
		for (int j = 0; j <= n; j++)
		{
			printf_s("%3d", L[i][j]);
		}
		printf_s("\n");
	}
}

void Handling(chuoi a, chuoi b, mang &L)
{
	chuoi c;
```

```
	int m = strlen(a) - 1;
	int n = strlen(b) - 1;


	CreateTable(L, m, n);

	for (int i = m - 1; i >= 0; i--)
	{
		for(int j = n - 1; j >= 0; j--)
		{
			if (a[i] == b[j]) L[i][j] = 1 + L[i + 1][j + 1];
			else L[i][j] = Max(L[i + 1][j], L[i][j + 1]);
		}
	}

	PrintItems(L, m, n);

	int i = 0, j = 0;
	int k = 0;

	while (i != m || j != n)
	{
		if (a[i] == b[j])
		{
			c[k] = a[i];
			k++;
			i++; j++;
		}
		else
		{

			if (L[i][j] == L[i+1][j]) i++;
			else j++;
		}
	}


	printf_s("\n\t\t KET QUA: ");

	FILE * fw;

	fopen_s(&fw, OUTPUT, "w");

	for (int i = 0; i < k; i++)
	{
		fprintf_s(fw, "%c", c[i]);
		printf_s("%c", c[i]);
	}

	fclose(fw);

}


int main()
{
```

```
        chuoi a, b;
        mang L;

        ReadFile(a, b);

        printf_s("Chuoi a: %s", a);
        printf_s("Chuoi b: %s", b);

        Handling(a, b, L);


        _getch();
    return 0;
}
```

Bài 14: BAO LÔ Cho n món hàng (n ≤ 50). Món thứ i có khối lượng là A[i] (số nguyên). Cần chọn những món hàng nào để bỏ vào một ba lô sao tổng khối lượng của các món hàng đã chọn là lớn nhất nhưng không vượt quá khối lượng M cho trước. (M ≤ 100). Mỗi món chỉ chọn 1 hoặc không chọn..

Dữ liệu đầu vào:
- Dòng thứ 1: Chưa hai số nguyên n và M.
- N dòng tiếp theo chứa trọng lượng W_i và giá trị V_i.

Dữ liệu đầu ra:
- Dòng thứ 1: Giá trị lớn nhất có thể
- Dòng thứ 2: Chứa chỉ số của các bao lô.

BAOLO.INP	BAOLO.OUT
5 11 3 3 4 4 5 4 9 10 4 4	11 5 2 1

```
#include <bits/stdc++.h>

using namespace std;
int n, m;

int W[50], V[50], L[50][50];

void ReadFile()
{
    ifstream f;
    f.open("BAOLO.INP");
    f >> n >> m;

    for(int i = 1; i <= n; i++)
    {
        f >> W[i] >> V[i];
    }
```

```
    sizeof(L, 0, sizeof(L)); // Khoi tao mang quy hoach dong
    f.close();
}

void Optimize() // Quy hoach dong
{
    for(int i = 1; i <= n; i++)
    {
        for(int j = 1; j<= m; j++)
        {
            L[i][j] = L[i - 1][j];
            if(j >= W[i] && L[i][j] < L[i - 1][j - W[i]] + V[i])
            {
                L[i][j] = L[i - 1][j - W[i]] + V[i];
            }
        }
    }
}

void Trace() // Truy vet
{
    cout << "Max value: " << L[n][m] << endl;
    cout << "selected packs : " << endl;

    while(n != 0)
    {
        if(L[n][m] != L[n - 1][m])
        {
            cout << "pack " << n << ": W = " << W[n] << " value = " << V[n] << endl;
            m = m - W[n];
        }
        n--;
    }
}

int main()
{
    ReadFile();
    Optimize();
    Trace();

    return 0;
}
```