

Feladat

Keressük meg a t négyzetes mátrixnak azt a fődiagonálissal párhuzamos átlóját, amelyben az elemek összege a legnagyobb!

Specifikáció

A feladat specifikálásához (és az eredmény megadásához) szükségünk van egy notációra, amivel az érintett átlókra hivatkozni tudunk. Ehhez az egész számokat fogjuk használni, 0-val jelölve a főátlót, és a pozitív számokkal sorban a főátló feletti átlókat, a negatívokkal a főátló alattiakat:

$$\begin{pmatrix} 0 & +1 & \cdots & +(n-1) \\ -1 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & +1 \\ -(n-1) & \cdots & -1 & 0 \end{pmatrix}$$

Így már megfogalmazhatjuk a feladatot, mint egy $-(n-1), \dots, (n-1)$ közötti egész keresése ($n \times n$ méretű mátrix esetén):

$$A = \mathcal{M}_{\mathbb{R}} \times_{t,i} \mathbb{Z} \times_{max,k} \mathbb{R} \times \mathbb{Z}$$

$$B = \mathcal{M}_{\mathbb{R}} \\ t'$$

$$Q = t' = t \wedge \exists n \in \mathbb{N} : t \in \mathbb{R}^{n \times n}$$

$$R = Q \wedge t \in \mathbb{R}^{n \times n} \wedge i \in \{-(n-1), \dots, (n-1)\} \wedge \\ \forall j \in \{-(n-1), \dots, (n-1)\} : S_D(t, j) \leq S_D(t, i)$$

ahol

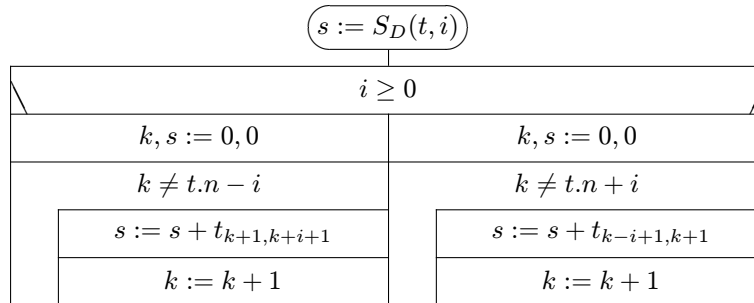
$$\mathcal{M}_T := \bigcup_{n \in \mathbb{N}^+} T^{n \times n}$$

$$S_D(t, i) := \sum_{k=1}^{n-|i|} \begin{cases} t_{k, k+i} & \text{ha } i \geq 0 \\ t_{k-i, k} & \text{egyébként} \end{cases}$$

Absztrakt program

A feladatot a maximumkeresés tételére vezetjük vissza, az S_D függvényt az $\{-(n-1), \dots, +(n-1)\}$ intervallumon használva, S_D -t pedig egy elágazásba ágyazott összegzéssel számoljuk ki, az alábbi programot kapva:

$i, k, max := -(t.n - 1), -(t.n - 1), S_D(t, -(t.n - 1))$	
$k \neq (t.n - 1)$	
$S_D(t, k + 1) \geq max$	
$i, max := k + 1, S_D(t, k + 1)$	SKIP
$k := k + 1$	



Fekete doboz-tesztelés

1. 1×1 -es mátrix
2. Azonos összegű átlókat tartalmazó mátrix
3. Tetszőleges mátrix

C++ implementáció

A C++ implementációban sokhelyen bejönnek ± 1 tagok, mivel a tömbök indexelése 0-val kezdődik, emiatt a mátrixoké is.

Programváz

```
// Fordítási direktívák

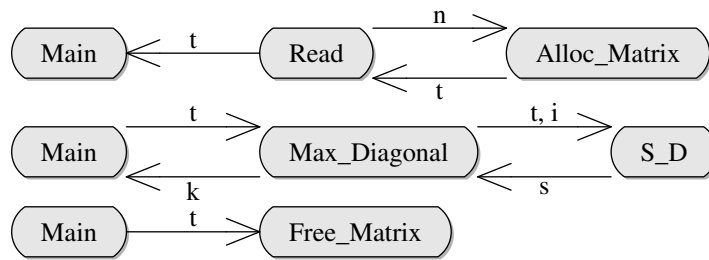
// Típusdefiníciók
// Függvénydeklarációk

int main (int argc, char **argv)
{
    // A főprogram
}

// Függvénydefiníciók
```

Adatforgalom a függvények között

A négyzetes mátrix sor- ill. oszlopszáma	$n \in \mathbb{N}^+$
A feldolgozandó mátrix	$t \in \mathbb{R}^{n \times n}$
Az éppen vizsgált átló	$i \in \{-(n-1), \dots, (n-1)\}$
Az aktuális átló összege	$s \in \mathbb{R}$
A legnagyobb összegű átló	$k \in \{-(n-1), \dots, (n-1)\}$



Alprogramok

Főprogram

FELADAT: A tesztelő keretrendszer része; elvégzi a beolvasást, a kiértékelést, és az erőforrások felszabadítását.

BEMENŐ ADATOK: A feldolgozandó mátrix: $t \in \mathcal{M}_{\mathbb{R}}$
 A legnagyobb összegű átló indexe: $k \in \{-(n-1), \dots, (n-1)\}$
 A std. kimenet: $out \in seq(ch)$

KIMENŐ ADATOK: A std. kimenet: $out \in seq(ch)$

TEVÉKENYSÉG: Meghívja a `Read` alprogramot, majd kiírja a std. kimenetre a `Max_Diagonal` függvény visszatérési értékét, végül felszabadítja a t által foglalt erőforrásokat a `Free_Matrix` alprogrammal.

```

int main (int argc, char **argv)
{
    Matrix t;

    Read (std::cin, t, true);

    std::cout << std::endl << "A legnagyobb_osszegu_atlo:_"
              << Max_Diagonal(t) << std::endl;

    Free_Matrix (t);

    return 0;
}

```

Read

FELADAT: A tesztelő keretrendszer része; beolvas, lefoglal, és visszaad egy mátrixot.

BEMENŐ ADATOK: A std. bemenet: $in \in seq(ch)$
 A mátrix mérete: $n \in \mathbb{N}$
 A lefoglalt mátrix: $t \in \mathbb{R}^{n \times n} \subset \mathcal{M}_{\mathbb{R}}$

KIMENŐ ADATOK: A std. bemenet: $in \in seq(ch)$
 A beolvasott mátrix: $t \in \mathcal{M}_{\mathbb{R}}$

TEVÉKENYSÉG: Beolvassa a std. bemenetről a négyzetes mátrix sor- ill. oszlopméretét, allokal egy ilyen méretű mátrixot, majd feltölti a std. bemenetről olvasott értékekkel.

```

void Read (std::istream &stream, Matrix &t, bool verbose)
{
    if (verbose) std::cout << "Matrix_merete:_" ;
}

```

```

int n;
stream >> n;
t = Alloc_Matrix (n);

if (verbose) std::cout << "Matrix_elemei" << std::endl;
for (int i = 0; i != t.n; ++i)
{
    for (int j = 0; j != t.n; ++j)
    {
        if (verbose) std::cout << "[" << i << ", " << j << "]: ";
        stream >> t.data[i][j];
    }
}
}

```

Alloc_Matrix

FELADAT: Létrehoz egy adott méretű négyzetes mátrixot reprezentáló adatszerkezetet.

BEMENŐ ADATOK: A négyzetes mátrix sor- ill. oszlopszáma: $n \in \mathbb{N}$

KIMENŐ ADATOK: A létrehozott, üres mátrix: $t \in \mathcal{M}_{\mathbb{R}}$

TEVÉKENYSÉG: Létrehoz egy `Matrix` típusú változót, feltölti annak n mezőjét, ezután a heap-en allokal n darab, egyenként n számot tartalmazni képes vektort, és ezeket eltárolja a változóban.

```

Matrix Alloc_Matrix (int n)
{
    Matrix t;

    assert (n >= 0);

    t.n = n;
    t.data = new Value*[n];
    for (int i = 0; i != n; ++i)
        t.data[i] = new Value[n];

    return t;
}

```

Free_Matrix

FELADAT: Felszabadítja egy mátrix reprezentációjához a heap-en lefoglalt erőforrásokat

BEMENŐ ADATOK: A felszabadítandó mátrix: $t \in \mathcal{M}_{\mathbb{R}}$

KIMENŐ ADATOK: A felszabadított mátrix: $t \in \mathcal{M}_{\mathbb{R}}$

TEVÉKENYSÉG: Végigmegy t sorain, és minden sorhoz felszabadítja az értékeit tartalmazó vektort; ezután a sorok vektorát is felszabadítja.

```

void Free_Matrix (Matrix &t)
{
    for (int i = 0; i != t.n; ++i)
        delete [] t.data[i];
    delete [] t.data;
}

```

S_D

FELADAT: Kiszámítja egy mátrix adott átlójának összegét

BEMENŐ ADATOK: A mátrix: $t \in \mathcal{M}_{\mathbb{R}}$
 Az átló indexe: $i \in \{-(n-1), \dots, (n-1)\}$

KIMENŐ ADATOK: Az átló összege: $s \in \mathbb{R}$

TEVÉKENYSÉG: Az összegzés tételét használja, a következő helyettesítésekkel:

$$m \leftarrow 0$$

$$n \leftarrow \begin{cases} t.n - i & \text{ha } i \geq 0 \\ t.n + i & \text{egyébként} \end{cases}$$

$$f(k) \leftarrow \begin{cases} t[k, k+i] & \text{ha } i \geq 0 \\ t[k-i, k] & \text{egyébként} \end{cases}$$

```
Value S_D (const Matrix &t, int i)
{
    Value s = 0;
    int k = 0;

    if (i >= 0)
    {
        while (k != t.n - i)
        {
            s += t.data[k][k + i];
            ++k;
        }
    } else {
        while (k != t.n + i)
        {
            s += t.data[k - i][k];
            ++k;
        }
    }

    return s;
}
```

Max_Diagonal

FELADAT: Kiszámítja egy mátrix legnagyobb összegű, főátlóval párhuzamos átlóját

BEMENŐ ADATOK: A mátrix: $t \in \mathcal{M}_{\mathbb{R}}$

KIMENŐ ADATOK: A legnagyobb összegű átló indexe: $i \in \{-(n-1), \dots, (n-1)\}$

TEVÉKENYSÉG: A maximumkeresés tételét használja, a következő helyettesítésekkel:

$$m \leftarrow -(t.n - 1)$$

$$n \leftarrow (t.n - 1)$$

$$f(k) \leftarrow S_D(t, k)$$

```
int Max_Diagonal (const Matrix &t)
{
```

```
int i = -(t.n - 1);
int k = i;
Value max = S_D (t, -(t.n - 1));

while (i != (t.n - 1))
{
    Value s = S_D (t, i + 1);

    if (s >= max)
    {
        k = i + 1;
        max = s;
    }

    ++i;
}

return k;
}
```

A program

```
#include <cassert>
#include <iostream>

typedef float Value;

struct Matrix
{
    int n;
    Value **data;
};

Matrix Alloc_Matrix (int n);
void Free_Matrix (Matrix &t);

Value S_D (const Matrix &t, int i);
int Max_Diagonal (const Matrix &t);

void Read (std::istream &stream, Matrix &t, bool verbose);

int main (int argc, char **argv)
{
    Matrix t;

    Read (std::cin, t, true);

    std::cout << std::endl << "A legnagyobb összegű atlo:_"
                << Max_Diagonal(t) << std::endl;

    Free_Matrix (t);

    return 0;
}

Matrix Alloc_Matrix (int n)
{
    Matrix t;

    assert (n >= 0);

    t.n = n;
    t.data = new Value*[n];
    for (int i = 0; i != n; ++i)
        t.data[i] = new Value[n];

    return t;
}

void Free_Matrix (Matrix &t)
{
    for (int i = 0; i != t.n; ++i)
        delete [] t.data[i];
    delete [] t.data;
}

Value S_D (const Matrix &t, int i)
```

```
{
    Value s = 0;
    int k = 0;

    if (i >= 0)
    {
        while (k != t.n - i)
        {
            s += t.data[k][k + i];
            ++k;
        }
    } else {
        while (k != t.n + i)
        {
            s += t.data[k - i][k];
            ++k;
        }
    }

    return s;
}

int Max_Diagonal (const Matrix &t)
{
    int i = -(t.n - 1);
    int k = i;
    Value max = S_D (t, -(t.n - 1));

    while (i != (t.n - 1))
    {
        Value s = S_D (t, i + 1);

        if (s >= max)
        {
            k = i + 1;
            max = s;
        }

        ++i;
    }

    return k;
}

// Testing framework starts here
void Read (std::istream &stream, Matrix &t, bool verbose)
{
    if (verbose) std::cout << "Matrix_merete:\n";

    int n;
    stream >> n;
    t = Alloc_Matrix (n);

    if (verbose) std::cout << "Matrix_elemei" << std::endl;
    for (int i = 0; i != t.n; ++i)
    {
        for (int j = 0; j != t.n; ++j)
        {
```

```
        if (verbose) std::cout << "[" << i << ", " << j << "]: ";
        stream >> t.data[i][j];
    }
}
```