ESCOLA DE PRIMAVERA DA MARATONA SBC DE PROGRAMAÇÃO

PROMOÇÃO:

CEFET-MG

UFMG

UFV

UNIFEI

USP Universidade de São Paulo

UFU

APOIO:

beecrowd

SBC Sociedade Brasileira de Computação

Grupo de Computação Competitiva

MATRIZES COM VECTOR<VECTOR>

Por: *Gabriel Couto Assis*

CONTEÚDOS

01 - PROBLEMA MOTIVADOR

Arnaldo e Bernardo são dois garotos que compartilham um peculiar gosto por curiosidades matemáticas. Nos últimos tempos, sua principal diversão tem sido investigar propriedades matemágicas de tabuleiros quadrados preenchidos com inteiros. Recentemente, durante uma aula de matemática, os dois desafiaram os outros alunos da classe a criar *quadrados mágicos*, que são quadrados preenchidos com números de 1 a N^2, de tal forma que a soma dos N números em uma linha, coluna ou diagonal principal do quadrado tenham sempre o mesmo valor. A *ordem* de um quadrado mágico é o seu número de linhas, e o *valor* do quadrado mágico é o resultado da soma de uma linha. Um exemplo de quadrado mágico de ordem 3 e valor 15 é mostrado na figura abaixo:

2	7	6
9	5	1
4	3	8

Para surpresa de Arnaldo e Bernardo, os outros alunos criaram um grande número de quadrados, alguns enormes, e alegaram que todos eram quadrados mágicos. Arnaldo e Bernardo agora precisam de sua ajuda, para verificar se os quadrados criados são realmente mágicos.

Você deve escrever um programa que, dado um quadrado, verifique se ele é realmente mágico.

Entrada

A primeira linha da entrada contém um único número inteiro N ($3 \leq N \leq 1000$.), indicando a ordem do quadrado (seu número de linhas). As N linhas seguintes descrevem o quadrado. Cada uma dessas linhas contém N números inteiros separados por um espaço em branco ($1 \leq$ valor de cada célula $\leq 10^9$).

Saída

Seu programa deve imprimir uma única linha. Caso o quadrado seja mágico, a linha deve conter o valor do quadrado (ou seja, a soma de uma de suas linhas). Caso contrário, a linha deve conter o número 0.

Exemplos de Entrada	Exemplos de Saída
3 1 1 1 1 1 1 1 1 1	0
4 16 3 2 13 5 10 11 8 9 6 7 12 4 15 14 1	34
3 4 8 9 11 7 3 6 5 10	0

02 - DEFINIÇÃO DO ALGORITMO

- Uma matriz é um array bidimensional.
- Se um vector representa uma “linha” de valores, um vector de vector representa uma linha de linhas.

```
vector<vector<int>> mat1(3,vector<int>(3));
```

- O vector de vector funciona como uma lista de listas.

```
// [[0,0,0],[0,0,0],[0,0,0]]
```

- Isso oferece a flexibilidade de manipular tanto o número de linhas quanto o número de colunas de forma independente.

```
// [0,0,0]
// [0,0,0]
// [0,0,0]
```

- Além disso, a STL oferece funções úteis como push_back(), resize(), e acesso eficiente a elementos com [].

03 - ALGORITMO

```cpp
int main() {
    vector<vector<int>> mat1(3,vector<int>(3));
    // [0,0,0]
    // [0,0,0]
    // [0,0,0]

    vector<vector<int>> mat2(3,vector<int>(3,-1));
    // [-1,-1,-1]
    // [-1,-1,-1]
    // [-1,-1,-1]

    mat1[2][2] = 1;
    // [0,0,0]
    // [0,0,0]
    // [0,0,1]

    mat1[0].push_back(2);
    // [0,0,0,2]
    // [0,0,0]
    // [0,0,1]

    for (int i = 0; i < mat1.size(); i++)
    {
        cout<<"[";
        for (int j = 0; j < mat1[i].size(); j++)
            cout<<mat1[i][j];
        cout<<"]\n";
    }

    return 0;
}
```

04 - FUNCIONAMENTO DO ALGORITMO

- Inicializa matriz vazia

```
vector<vector<int>> mat1(3,vector<int>(3));
// [0,0,0]
// [0,0,0]
// [0,0,0]
```

- Acessa posição matriz

```
mat1[2][2] = 1;
// [0,0,0]
// [0,0,0]
// [0,0,1]
```

- Utiliza funções de vector

```
mat1[0].push_back(2);
// [0,0,0,2]
// [0,0,0]
// [0,0,1]
```

- Inicializa matriz preenchida

```
vector<vector<int>> mat2(3,vector<int>(3,-1));
// [-1,-1,-1]
// [-1,-1,-1]
// [-1,-1,-1]
```

- Printa matriz

```
for (int i = 0; i < mat1.size(); i++)
{
    cout<<"[";
    for (int j = 0; j < mat1[i].size(); j++)
        cout<<mat1[i][j];
    cout<<"]\n";
}
```

05 - RESOLUÇÃO DO PROBLEMA MOTIVADOR

1. Crie a matriz e armazene o quadrado mágico
2. Teste se existe apenas um de cada
3. Some os valores da diagonal principal
4. Verifique se todas as linhas e colunas obedecem a soma correta
5. Caso esteja tudo certo, imprima o valor da soma mágica

5.1 - ARMAZENAR, VERIFICAR REPETIÇÃO E SOMAR

```cpp
// Pega o tamanho do quadrado
int n;
cin >> n;

// Criando a matriz usando vector de vector
vector<vector<int>> quadrado(n, vector<int>(n));
set<int> contador; // Para contar a ocorrência de cada número
int magico = 0;

for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
        cin >> quadrado[i][j];

        // Soma a diagonal principal
        if (i == j) {
            magico += quadrado[i][j];
        }
        contador.insert(quadrado[i][j]);
    }
    if (contador.size()!=(i+1)*n) {
        cout << "0\n";
        return 0;
    }
}
```

```cpp
// Pega o tamanho do quadrado
int n;
cin >> n;

// Criando a matriz usando vector de vector
vector<vector<int>> quadrado(n, vector<int>(n));
unordered_map<int, int> contador; // Para contar a ocorrência de cada número
int magico = 0;

for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
        cin >> quadrado[i][j];

        // Soma a diagonal principal
        if (i == j) {
            magico += quadrado[i][j];
        }

        // Conta as ocorrências de cada número
        if (contador[quadrado[i][j]] > 0) {
            cout << "0\n";
            return 0;
        } else {
            contador[quadrado[i][j]]++;
        }
    }
}
```

5.2 - VERIFICA SOMA EM COLUNAS E LINHAS

```cpp
// Verifica as somas das linhas e colunas
for (int i = 0; i < n; i++) {
    int somal = 0, somac = 0;

    for (int j = 0; j < n; j++) {
        somal += quadrado[i][j]; // Soma da linha
        somac += quadrado[j][i]; // Soma da coluna
    }

    // Verifica se a soma da linha e da coluna é igual à soma mágica
    if (somal != magico || somac != magico) {
        cout << "0\n";
        return 0;
    }
}

cout << magico << endl;
return 0;
```

06 - OUTRAS APLICAÇÕES

- **Grafos**: Representação de grafos através de listas de adjacência, onde cada nó do grafo possui uma lista de nós conectados.
- **Tabelas Dinâmicas**: Aplicações que requerem a criação de tabelas de tamanho variável, como planilhas ou sistemas de armazenamento de dados.
- **Simulação de Jogos**: Matrizes são comumente usadas para representar tabuleiros ou mapas em jogos, onde cada célula da matriz representa uma posição ou estado no jogo.

07 - PROBLEMAS

2407 - https://judge.beecrowd.com/pt/problems/view/2407 (Quadrado Mágico)

2450 - https://judge.beecrowd.com/pt/problems/view/2450 (Matriz Escada)

OBRIGADO PELA ATENÇÃO

Grupo de Computação Competitiva

GCC.CEFETMG

gcc-cefetmg.github.io/Site-GCC

@gcc.cefetmg