Problema

Dada uma matriz binária 2D de tamanho MxN preenchida com '0' (zero) e '1' (um), encontre o retângulo de maior área contendo apenas '1' e retorne o valor de sua área. Exemplo:

Entrada:

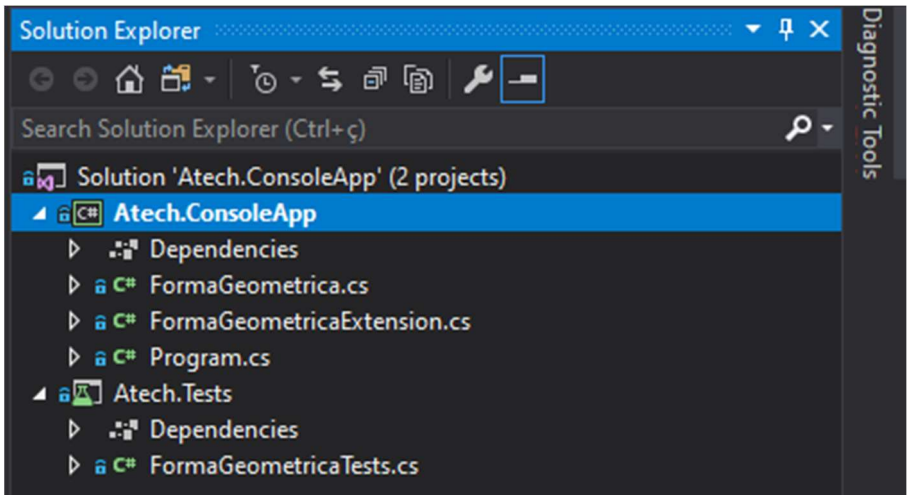
[

['1','0','1','0','0'],
['1','0','1','1','1'],
['1','1','1','1','1'],
['1','0','0','1','0']
]

Saída: 6

Solução

Criada uma solução em DotNet core 2.2 do tipo console, para validar dados de entrada e informar o maior valor de retângulo encontrado.

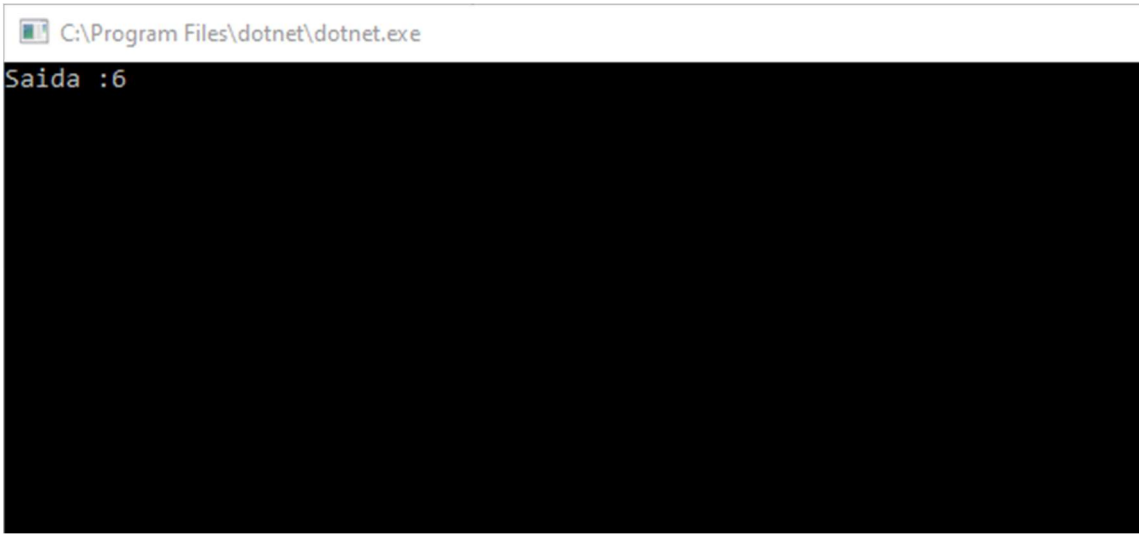


Program.cs

Arquivo de inicialização do programa, armazena matriz na variável “entrada” para executar o processamento do método “RetornarFormaGeometrica”.

```
class Program
{
    static void Main(string[] args)
    {
        int[,] entrada = new int[4, 5]
        {
            {1,0,1,0,0},
            {1,0,1,1,1},
            {1,1,1,1,1},
            {1,0,0,1,0}
        };

        var resultado = entrada
            .RetornarFormaGeometrica();

        var textoApresentacao = "Nenhum retângulo válido.";

        if (resultado.Valido)
            textoApresentacao = string.Concat("Saida :", resultado.Area);

        Console.WriteLine(textoApresentacao);
        Console.ReadKey();
    }
}
```

Após o processamento valida o resultado e mostra a mensagem no console.



FormaGeometrica.cs

Arquivo que a estrutura da forma geométrica.

```
namespace Atech.ConsoleApp
{
    public class FormaGeometrica
    {
        public FormaGeometrica()
        {
            Vertice = 1;
        }
        public int ColunaInicial { get; set; }
        public int Area { get; set; }
        public bool Valido { get; set; }
        public int Vertice { get; set; }
    }
}
```

FormaGeometricaExtension.cs

Arquivo responsável por fazer todo a análise de sequências válidas dentro da matriz.

```
public static class FormaGeometricaExtension
{
    const int COLUNA_INICIAL = 0;
    const int VERTICE_VALIDO = 1;
    const int INCREMENTO_LINHA = 1;
    const int AREA_VALIDA = 2;

    public static FormaGeometrica RetornarFormaGeometrica(this int[,] numeros)...

    private static FormaGeometrica AnalisarLinha(this int[,] numeros, int indexLinha, int colunaInicial)...

    private static void SeValidoIncrementarObjeto(this FormaGeometrica resultado, FormaGeometrica forma)...

    private static void ValidarVertice(this FormaGeometrica arestaAtual, int indexColuna, int vertice)...

    private static void ExecutarRecursividade(this int[,] numeros, int indexLinha, FormaGeometrica aresta)...
}
```

Contém como método público “RetornarFormaGeometrica”,

```
public static FormaGeometrica RetornarFormaGeometrica(this int[,] numeros)
{
    var resultado = new FormaGeometrica();

    for (int indexLinha = 0; indexLinha < numeros.GetLength(0); indexLinha++)
    {
        var forma = numeros
            .AnalisarLinha(indexLinha, COLUNA_INICIAL);

        resultado
            .SeValidoIncrementarObjeto(forma);
    }

    resultado.Valido = resultado.Vertice >= AREA_VALIDA;
    return resultado;
}
```

Que tem como responsabilidade de percorrer as linhas da matriz e fazer a chamada para o armazenamento do maior resultado encontrado no método “SeValidoIncrementarObjeto”.

```
private static void SeValidoIncrementarObjeto(this FormaGeometrica resultado, FormaGeometrica forma)
{
    if (forma.Valido && resultado.Area <= forma.Area)
    {
        resultado.Area = forma.Area;
        resultado.ColunaInicial = forma.ColunaInicial;
        resultado.Valido = forma.Valido;
        resultado.Vertice = forma.Vertice;
    }
}
```

Para validar a linha utilizamos o método “AnalisarLinha”

```
private static FormaGeometrica AnalisarLinha(this int[,] numeros, int indexLinha, int colunaInicial)
{
    var arestaAtual = new FormaGeometrica();

    for (int indexColuna = colunaInicial; indexColuna < numeros.GetLength(1); indexColuna++)
    {
        var vertice = numeros[indexLinha, indexColuna];

        arestaAtual
            .ValidarVertice(indexColuna, vertice);
    }

    numeros
        .ExecutarRecursividade(indexLinha, arestaAtual);

    return arestaAtual;
}
```

Que é responsável por percorrer todas as colunas da linha em questão e passar o valor encontrado para o método “ValidarVertice”.

```
private static void ValidarVertice(this FormaGeometrica arestaAtual, int indexColuna, int vertice)
{
    if (vertice == VERTICE_VALIDO)
    {
        arestaAtual.Area += vertice;

        if (!arestaAtual.Valido)
            arestaAtual.ColunaInicial = indexColuna;

        arestaAtual.Valido = true;
    }
    else
    {
        arestaAtual.Area = 0;
        arestaAtual.Valido = false;
        arestaAtual.ColunaInicial = COLUNA_INICIAL;

    }
}
```

Se cada valor for valido (vertice = 1), inicia a sequência para determinar uma linha válida, caso contrário reinicia os valores.

Por último temos o método “ExecutarRecursividade”

```
private static void ExecutarRecursividade(this int[,] numeros, int indexLinha, FormaGeometrica aresta)
{
    aresta.Valido = aresta.Area > AREA_VALIDA;
    int indexProximaLinha = indexLinha + INCREMENTO_LINHA;
    bool exiteProximaLinha = indexProximaLinha < numeros.GetLength(0);
    if (aresta.Valido && exiteProximaLinha)
    {
        var proximaAresta = AnalisarLinha(numeros, indexProximaLinha, aresta.ColunaInicial);
        aresta.Area += proximaAresta.Area;
        aresta.Valido = proximaAresta.Area > AREA_VALIDA;

        if (aresta.Valido)
            aresta.Vertice += VERTICE_VALIDO;
    }
}
```

Sua responsabilidade é a de em caso de existir próximas linhas, refazer todos os passos, até chegar ao final da matriz ou encontrar uma linha inválida.

Testes

Criado o projeto de testes utilizando XUnit.

Foram validados três tipos de entradas

1 - Matriz 4x5

Entrada = new int[4, 5] { {1,0,1,0,0}, {1,0,1,1,1}, {1,1,1,1,1}, {1,0,0,1,0} };

Saída: 6

2 - Matriz 8x5

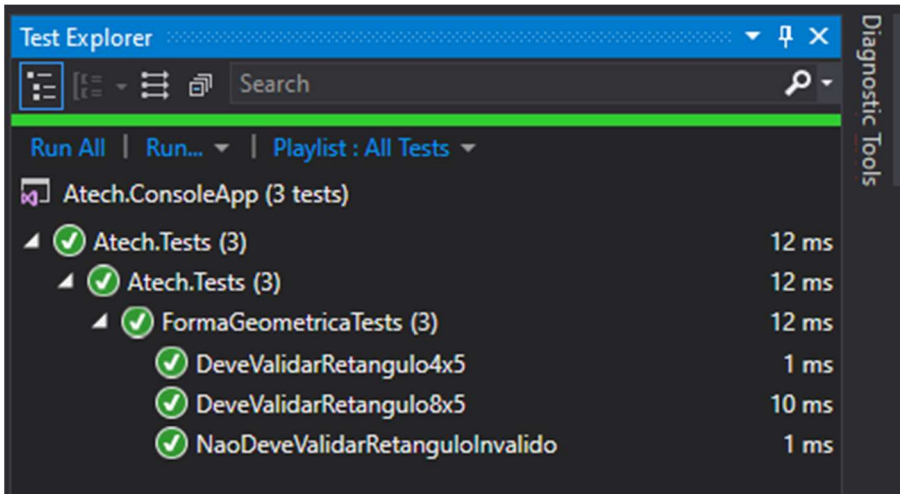
Entrada = new int[8, 5] { {1,0,1,0,0}, {1,0,0,1,1}, {1,0,0,1,1}, {1,0,0,1,0}, {1,0,1,0,0}, {1,0,1,1,1}, {1,1,1,1,1}, {1,0,0,1,0} };

Saída: 6

3 - Matriz 2x5

Entrada = new int[2, 5] { {1,0,1,0,0}, {1,0,1,1,1} };

Saída: Nenhum retângulo válido.



Conclusão

O programa recebe a entrada no formato de matriz, semelhante ao exemplo informado no início do documento.

Deve-se percorrer todas as linhas da matriz, validando se existe sequência válida (A1).

Foi utilizado recursividade ao encontrar uma linha válida e executado o mesmo processo de validação em todas as linhas posteriores, o intuito é encontrar o retângulo e calcular sua área (A1).

A1 - Sequencia válida

Para formar um retângulo válido, linhas e colunas não podem ter o mesmo valor, no início do documento foi demonstrado o valor 6 como saída válida, constituindo 2 linhas x 3 colunas.

GitHub

https://github.com/antoniofcpinheiro/Atech