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']
]
SAIDA: 6
```

RESOLUÇÃO

Para a entrada dos dados, o algoritmo irá pedir o número de linhas e colunas que a matriz deverá ter, reservar a memória necessária para a matriz e posteriormente solicitar a entrada dos valores de cada coluna dentro de cada linha, até que todas linhas e colunas tenham sidas preenchidas.

Por se tratar de uma matriz binária, durante a entrada dos dados, somente são esperados os valores ‘1’ e ‘0’, consequentemente o algoritmo ao identificar qualquer outra entrada a não ser estas, finaliza com erro.

Sendo todas a entradas validas, foi optado por realiza a conversão de caracteres de texto para o formato numérico, desta forma armazenando uma matriz de fato booleana e não uma de caracteres ‘1’ e ‘0’. Assim podemos alocar um novo espaço de memória, realizar a conversão dos caracteres para o formato numérico, desalocar o espaço da matriz antiga e nos próximos passos utilizar esta nova matriz.

Com toda a entrada e preparação dos dados realizada, podemos partir para a solução do problema, foi abortado um método sequencial que realiza a verificação de todas as posições que contenham o valor ‘1’, percorrendo da direita para a esquerda e de cima para baixo, desta forma computando todas as áreas possíveis, verificando sempre qual a maior área e a armazenando.

Para calcular uma área primeiro verificasse em qual a direção, para a direita ou para baixo, aquela posição terá o maior número valores um.

Exemplo:

$$Matriz = \begin{vmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{vmatrix}$$

Posição	Alcance linha	Alcance coluna
[0,0]	1	5
[0,2]	1	3
[1,0]	1	3
[1,2]	3	2
[2,0]	5	2
[2,1]	4	1
[2,2]	3	1
[2,3]	2	2
[2,4]	1	1
[3,0]	1	1
[3,3]	1	1

Identificado a direção com o maior número de posições, podemos utilizar ela com base para a construção do retângulo daquela posição, base e altura serão armazenados em duas variáveis, inicializadas com o valor um, que é a menor área possível ao identificar um possível retângulo. O algoritmo agora irá realizar uma escolha, caso tenha um maior número de posições na linha ele irá seguir analisando linha por linha e o valor da base será o alcance calculado, caso contrário, ele irá seguir analisando coluna por coluna e o valor da altura será o alcance calculado.

Independente da direção tomada, a análise é a mesma, a cada linha ou coluna verificada a partir daquela posição, computamos o seu alcance, ou seja, o número de posições com o valor um que foi verificado. Desta forma podemos utilizar este número como contador dentro de uma iteração, na qual a cada iteração verificamos se o alcance naquela direção, na próxima linha ou coluna, se manteve o mesmo, se está maior ou se diminuiu, até que todo o alcance possível seja averiguado.

Quando o alcance aumenta ou se mantem, no caso de uma verificação por linha, computamos que a altura aumentou, e no caso de uma verificação por coluna que a base aumentou. Se o alcance diminuir o algoritmo sinaliza para as próximas iterações que houve diminuição e realiza uma análise se esta diminuição deve ser contabilizada, ou seja, se esta diminuição naquela verificação gera uma área maior do que a já averiguada até ali, em caso positivo ele atualiza a base e a altura.

Exemplo:

Posição	Alcance linha	Alcance coluna
[1,2]	3	2

Partindo da posição [1,2] o algoritmo seguiria a interação pelas linhas, já que foi apresentando um maior alcance. Desta forma a próxima posição seria verificar o alcance da linha abaixo [2,2], que seria de três também, mantando a base em três e aumentando a altura para dois. Não haveria próxima iteração, já que o valor de [3,2] seria o valor zero, com isso fazendo o algoritmo terminar as iterações e verificar se a área calculada (3x2) seria maior que o alcance inicial (3), retornando assim a área de valor seis.

Caso ocorra de o sinalizador de diminuição ser ativado, nas próximas iterações o algoritmo não irá averiguar o caso de tamanho igual ou maior, agora todas as iterações serão tratadas como menores, e a cada iteração será verificado se houve uma nova diminuição, assim evitando que seja computado uma área errada.

O sinalizador de diminuição contem nele o valor do novo alcance computado, desta forma podemos utilizar ele para formar novas áreas e realizar as verificações necessárias a fim de encontrar uma maior área.

Exemplo:

Posição	Alcance linha	Alcance coluna
[2,0]	5	2

Partindo da posição [2,0] o algoritmo seguiria a interação pelas linhas, já que foi apresentando um maior alcance. Desta forma a próxima posição seria verificar o alcance da linha abaixo [3,0], que seria de um, ocasionando na redução da base, desta forma o algoritmo iria verificar se esta nova área (2x1) seria maior que a área anterior (5x1), o que seria falso, consequentemente não alterando o valor da área e sinalizando a diminuição de base, influenciando nas próximas iterações, fazendo com que o retorno fosse área cinco.

Contudo existem casos particulares que devem ser analisados, para otimizar o algoritmo. O primeiro deles é a possibilidade de que o retângulo se forme diretamente na linha ou coluna analisada, ou seja, altura ou largura igual a um, sendo facilmente verificado quando o a direção contraria a maior identificada é igual a um.

Exemplo:

Posição	Alcance linha	Alcance coluna
[0,0]	1	5

Nesta situação o algoritmo já retorna a área calculada como o número de posições da maior direção. Outro caso é que mesmo ao final de todas as verificações, a área armazenada ainda seja menor do que a área formada pelo retângulo de altura ou tamanho um, conforme demonstrando no exemplo anterior, neste caso o algoritmo retorna assim como no outro caso o maior alcance.

Assim ao final da iteração teremos a maior área possível dentro da analise daquela posição, podendo retornar o valor computado e passar para a próxima posição. Desta forma obtendo a maior área possível dentro da matriz informada.

Exemplo de execução do algoritmo: