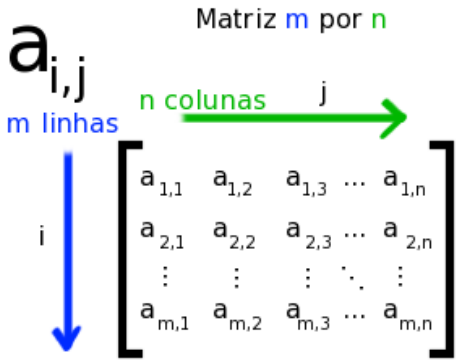
Area do maior retangulo contendo ‘1’ dentro de uma matrix binária MxN:

Inicialmente, é preciso saber qual será o matriz usada no algoritmo, por se tratar de uma matriz sem as dimensões previamente definidas, é pedido para o usuário escolher se ele quer entrar apenas com os valores de linha e coluna (m e n, representados na figura abaixo) ou com todos os valores, ou seja, quatidade de linhas, de colunas e todos os elementos dessa matriz (todos os “a” da figura abaixo). Tudo isso acontece até a linha 139, no método “defineMatrix”.



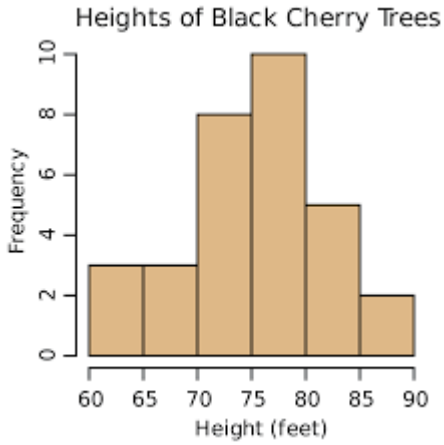
Com as dimensões criadas da nossa matriz é preciso popular cada um de seus elementos (“a”), se o usuário escolher o preenchimento aleatório (representado pela variável “manual”, no caso com o valor “false”), o método “populateMatrix” irá preencher com 0 ou 1 (por ser uma matriz binária, são os únicos valores possíveis) cada elemento, percorrendo uma linha de cada vez até a última coluna. Por outro lado, se o usuário preferir entrar manualmente com os valores, será pedido pra ele digitar 0 ou 1 para cada elemento, sendo exibido na tela qual posição está sendo preenchida, se caso ele digitar algum diferente, o algoritmo lança uma mensagem de erro e pede para digitar novamente. Toda essa parte acontece com a execução da linha 141. no método “defineMatrix”.

Com nossa matrix criada e preenchida, ela é exibida na tela (linha 142, no método “defineMatrix”). Para fins de exemplificação, vamos considerar que essa foi matrix a criada:

1	0	1	0	0
1	0	1	1	1
1	1	1	1	1
1	0	0	1	0

Feito isso o método “defineMatrix” retorna para a “main” com a variável “matrix” preenchida de acordo com a imagem a cima, então, o método “findRectMaxArea” é chamado tendo nossa matriz como valor de referência.

Agora vamos entrar na parte principal do algoritmo, em que inicia-se os cálculos para encontrar a área do maior retangulos de 1. Vamos usar como base histogramas, gráficos de barras verticais:



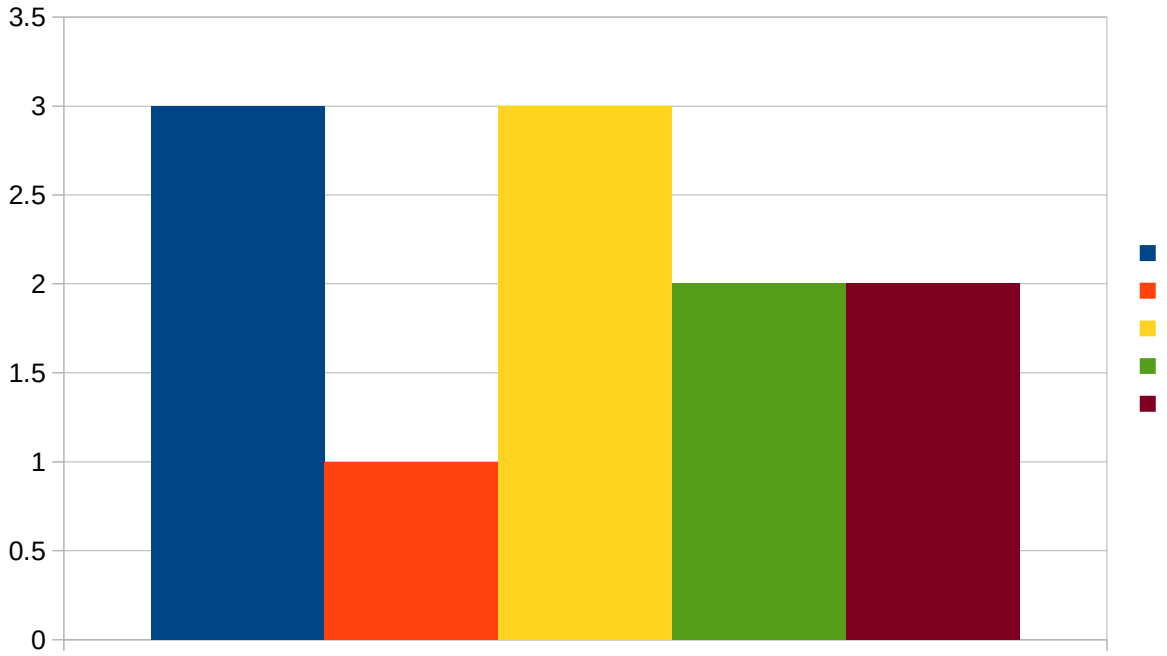
Dessa forma vamos calcular “barras verticais” para cada linha da nossa matriz, usando sempre a linha de “cima” como base. Dessa forma inicia-se percorrendo toda matriz a partir da segunda linha, pois como a primeira não teria uma linha “acima” seu valor se mantem igual. Todo elemento é verificado, se ele for igual a 1, ele é somado com seu “vizinho superior”, mas se ele for igual a 0, o valor continua zero. Isso é feito para todas as linhas da matriz, seguindo nossa matriz de exemplo, ele ficaria da seguinte forma:

1	0	1	0	0
2	0	2	1	1
3	1	3	2	2
4	0	0	3	0

Com essa matriz resultante, é como se tivéssamos agora um histograma para cada linha, e cada barra tendo como altura o valor de cada elemento. Dessa forma, encontramos as “maiores barras” constantes, ou seja, sem nenhum “buraco” (zero) entre os valores.

Assim, cada linha agora possui uma área, deve-se então ser calculada para encontrarmos o valor máximo dentro da matriz, é chamado então o método “calculateMaxArea” que recebe como parâmetro cada linha da matriz.

Esse cálculo, pode parecer um pouco difícil mas não é, cada linha é percorrida elemento a elemento, com a inteção de encontrar valores “mínimos” entre pares ou grupos de barras sequenciais e para cada desses grupos, a área é calculada pela operação base x altura, sendo a base a quantidade de barras com valores “mínimos” e a altura o próprio valor mínimo. Com o valor da aŕea calculada, ele é sempre comparado ao anterior afim de encontrar o maior, assim como com todos os elementos. Vamos usar como exemplo, a linha 3 da nossa matriz, seu histograma ficaria da seguinte forma:



Fixando o primeiro elemento (barra azul) e percorrendo os outros, vimos que o valor minimo entre todos eles é 1, o valor da barra laranja, dessa forma a altura do nosso retangulo é 1 e a largura é 5 (quantidade de barras com o valor mínimo), então a área é 5 e como a área do primeiro elemento sozinho (3 x 1) é menor que 5, nossa aŕea maxima agora é 5.

Seguindo para o próximo elemento (barra laranja), e fazendo mesma lógico, encontramos a área de valor 4, como é menor que nosso valor máximo até agora, nossa área máxima continua sendo 5.

Terceiro elemento (barra amarela), nessa caso vimos que ao percorrer todos os elementos restantes a partir dele, vimos que o valor mínimo é 2 (altura) e a largura é 3 (quantidade de barras sequenciais com o valor minimo), dessa forma a área é igual a 6, que é maior que nossa área maxima atual, portanto, nossa área máxima agora é 6.

Usando da mesma lógica para os elementos restantes, vimos que nossa área maxima continua sendo 6. Fazendo o mesmo processo para todas as outras linhas, vimos que nossa área maxima é 6, como podemos ver se analisarmos nossa matriz inicial:

1	0	1	0	0
1	0	1	1	1
1	1	1	1	1
1	0	0	1	0

No final da execução do algoritmo, o valor máximo é exibido na tela.

Complexidade

Quando analisamos a complexidade desse algoritmos vimos que ele depende da quantidade de colunas da matriz escolhida, sendo:

O (colunas)