Uma Possível Solução ao Problema do Maior Retângulo Livre em um Campo Minado

Carlos André Sousa Rodrigues

carlos.sousa@acad.pucrs.br

Faculdade de Informática – PUCRS

Porto Alegre – Brasil

10 de setembro de 2017

Resumo

Este artigo apresenta uma possível eficiente solução ao primeiro problema proposto na disciplina de Algoritmos e Estruturas de Dados II, no semestre 2017/02, cujo objetivo é desenvolver um algoritmo capaz de encontrar a maior área retangular possível livre em um campo minado, a partir da largura, altura, e quantidade de minas presentes neste campo, acompanhado da posição de cada uma. São apresentadas brevemente duas possíveis estratégias que poderiam ser usadas para a resolução, sendo uma extremamente mais eficiente que a outra, ao passo que uma solução concreta da melhor estratégia é explicada em detalhes, apresentando o resultado de alguns casos de teste.

Introdução

Para definir um campo minado, é dado como entrada um arquivo texto contendo todos os dados necessários para sua representação. Este arquivo é composto basicamente por duas partes:

1. Na primeira linha, 3 números são informados (w h m), sendo o primeiro a largura do campo (variando de 1 à w), seguido da altura do campo (também variando de 1 à h), e por último, a quantidade de minas terrestres presentes no campo.

2. Da segunda linha em diante, são apresentados dois números (x y) que representam as posições das m minas dispostas no campo, sendo x um número que varia de 1 à h, e y variando de 1 à w. De maneira similar, podemos interpretar x e y como sendo a linha e a coluna de determinada mina, respectivamente.

O desafio é encontrar a maior área retangular livre de minas terrestres no campo apresentado, informando sua localização e área. Alguns casos de teste foram propostos, que variam de 10.000 w **x** 10.000 h com 100 minas, à 100.000 w **x** 100.000 h com 1.000 minas. Todos os casos serão apresentados detalhadamente nas seções seguintes.

Estratégia 1

Este problema poderia ser resolvido de uma maneira pouco eficiente, sendo talvez o tipo de

solução mais intuitiva ao nos depararmos com algum problema assim. O plano aqui seria considerar nosso campo minado como sendo uma grade, que incorporaria linhas e colunas bem definidas, e suas coordenadas, minadas ou não. Para sua possível implementação física, poderiam ser utilizadas estruturas como uma matriz, ou apenas uma relação numérica para iterar sobre todos os espaços (livres ou não), dentre várias outras possíveis. O fato é que aqui estaríamos interessados em considerar toda a estrutura do campo, tanto os espaços livres quanto os espaços minados, de maneira física. Aqui podemos nos deparar com dois principais problemas:

1. Memória: ao tentarmos, por exemplo, utilizar uma matriz para guardar nosso campo minado, deve-se levar em consideração o tamanho, o que para pequenos campos funcionaria perfeitamente, porém, para os casos que apresentam campos maiores, como os casos testes que serão discutidos, essa estratégia não é adequada.

2. Processamento: para a busca do maior retângulo livre, provavelmente teríamos de, no mínimo, iterar uma vez sobre a representação de cada célula do campo. Considerando o tamanho dos campos em que estamos interessados, na melhor das hipóteses esse processo ainda seria custoso em termos de processamento.

Finalizando, podemos concluir que se estamos interessados em analisar a complexidade (O, Ω, Θ) de algum algoritmo que utilize essa estratégia, os N elementos a serem levados em consideração seria igual à área total do campo, o que, por exemplo, no caso de teste 100.000 x 100.000, N seria 10^{10}.

Estratégia 2

Felizmente, existem maneiras mais eficientes de se resolver o problema proposto, ao qual discutiremos neste tópico.

Analisando os campos minados dos casos de teste, é possível ver a clara diferença de proporção entre as células livres e as células minadas. Todos os casos, em conjunto, obedecem a uma regra de proporção. A medida que avançamos do primeiro ao último caso teste, apesar da dimensões do campo e o número de minas crescerem aritmeticamente (mantendo certa proporção entre si), a área total resultante do produto das duas dimensões cresce de maneira quadrática. Logo, a proporção entre o número de minas e as células livres aumenta, neste universo, exatamente em 1.000.000 a cada teste.

Após essa breve observação, temos um recurso extremamente importante a ser aproveitado: o número de células cresce quadraticamente, mas o número de minas cresce de maneira linear. Como estamos preocupados em uma solução eficiente para grandes campos minados, e sabendo que o número de minas é bem menor que o número de células livres, podemos manter o foco sobre as minas presentes no campo, tirando proveito de sua localização.

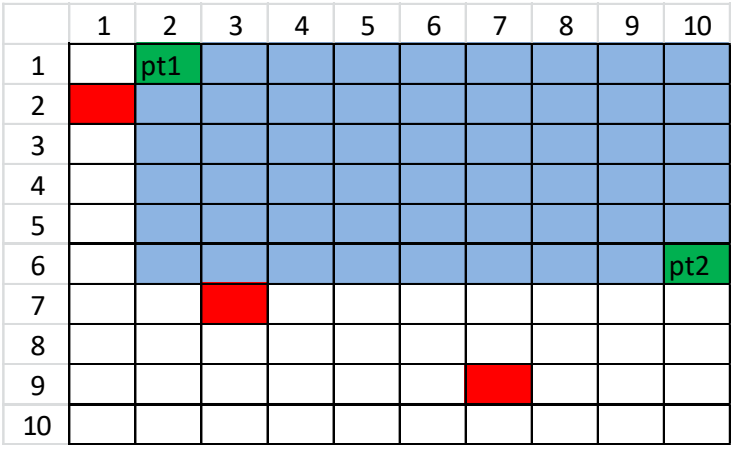
Nosso interesse aqui seria armazenar apenas as células minadas em alguma estrutura, e a partir disso, construir um algoritmo que analise as razões matemáticas entre as minas e as delimitações do campo (w e h) para enumerar todos os retângulos livres, e ao fim, encontrar aquele de maior área. Essa foi a estratégia utilizada para a construção da solução proposta neste artigo.

Solução

Tendo em vista o formato dos dados de entrada, e os fatores discutidos na estratégia 2, podemos ler estes dados e armazena-los em alguma estrutura, aplicando algumas operações sobre os mesmos a fim de extrair a informação que nos interessa.

A seguinte solução foi implementada utilizando a linguagem Java, porém o código será aqui adaptado em pseudocódigo para a simplificação e exposição mais direta do que foi desenvolvido.

Inicialmente, é interessante definir como a informação relativa à localização do maior retângulo será apresentada. Para isso, utilizaremos dois pontos, que juntos delimitarão o retângulo: o ponto esquerdo (pt1x, pt1y), e o ponto direito (pt2x, pt2y), em que um deve estar na extremidade de cima, e o outro na debaixo. Lembrando que x corresponde à linha e y à coluna nos exemplos a serem mostrados neste artigo. Portanto, um retângulo livre pode ser delimitado como se segue:



Onde:

pt1x: 1 pt2x: 6

pt1y: 2 pt2y: 10

Sabendo disso, podemos utilizar dois objetos para representar as minas terrestres e as áreas retangulares livres:

```
Mina
{
	x;
	y;
}

RetanguloLivre
{
	pt1x;
	pt1y;
	pt2x;
```

```
	pt2y;
	area;

	metodo calculaArea {
		area =  (pt2x-pt1x+1) * (pt2y-pt1y+1);
	}
}
```

No programa principal, enumeremos oito “variáveis globais” principais que utilizaremos:

h - Altura do campo

w - Largura do campo

pt1x - Linha do ponto esquerdo de um determinado retângulo.

pt1y - Coluna do ponto esquerdo de um determinado retângulo.

pt2x - Linha do ponto direito de um determinado retângulo.

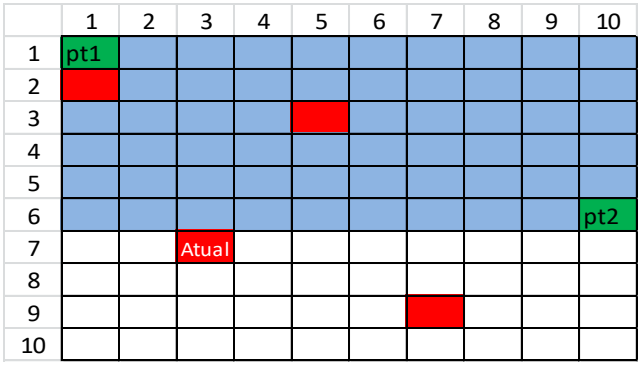
pt2y - Coluna do ponto direito de um determinado retângulo.

minas – Uma lista (ArrayList, no Java) contendo todos os objetos “Mina” presentes no campo.

retangulos – Uma lista (ArrayList, no Java) contendo todos os objetos “RetanguloLivre” que encontraremos.

Após a definição das estruturas que utilizaremos, podemos ler os dados de um arquivo válido, guardando as dimensões do campo em h e w, e uma lista de objetos “Mina” em minas. Após armazenar os dados, a lista de minas deve ser ordenada, onde o primeiro critério seja a linha (x), e o segundo, a coluna (y), ambos em ordem crescente. Assim podemos estabelecer um padrão ao realizar a iteração entre as minas. Para a implementação real dessa ordenação, foram utilizados alguns recursos da linguagem Java, e por se tratar de algo trivial e de pouca relevância ao objetivo proposto, seu pseudocódigo será omitido deste artigo.

Para encontrar o maior retângulo, iremos primeiramente encontrar todos os retângulos livres candidatos a serem o de maior área, e para cada um que encontrarmos, armazenar um objeto “RetanguloLivre” com suas propriedades na nossa lista de retângulos. Para realizar essa tarefa, iremos iterar sobre cada mina terrestre disposta no campo, e inicialmente para cada um dos 4 lados em relação à mina atual, delimitar o maior retângulo possível, ignorando inicialmente as outras minas terrestres, como segue abaixo:



Note que o campo inicial está delimitado com os pontos pt1 e pt2, seguindo a seguinte regra de atribuição:

```
pt1x = 1;
pt1y = 1;
pt2x = atual.x - 1;
pt2y = w;
```

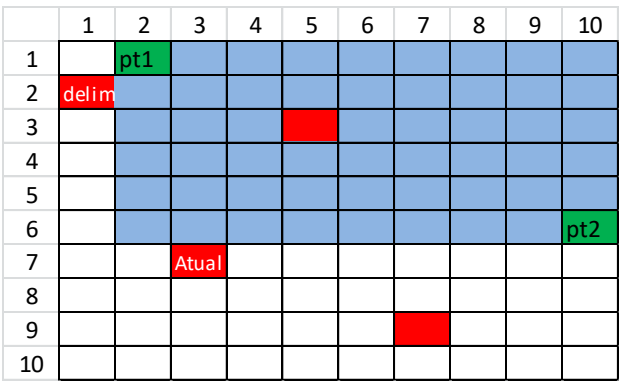
Com isso, podemos iniciar uma nova iteração sobre a lista de minas, delimitando nosso retângulo por cada uma das minas terrestres, de duas maneiras distintas, o que nos possibilita extrair dois diferentes retângulos acima dessa mina terrestre. Essas duas maneiras podem ser representadas em pseudocódigo. O seguinte trecho expõe com mais completude este processo:

- Delimitação 1

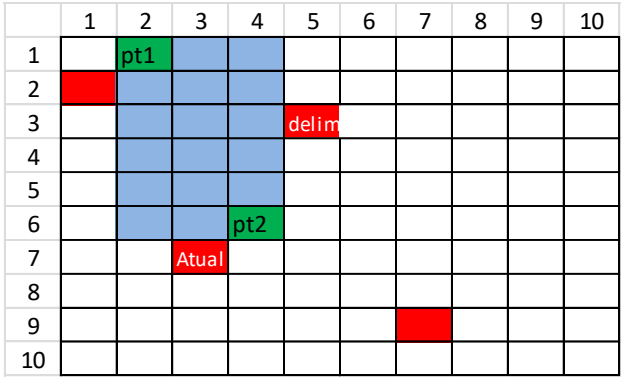
```
Para todo atual de minas {
  Se (atual.x > 1)
      pt1x = 1;
      pt1y = 1;
      pt2x = atual.x - 1;
      pt2y = w;

      Para todo delim de minas {
            Se (delim != atual && delim.x < atual.x) {
                  Se (delim.y < atual.y && delim.y >= pt1y
                      && delim.x >= pt1x) {
                        Pt1y = delim.y + 1;
                  } senão
                  Se (delim.y > atual.y && delim.y <= pt2y
                      && delim.x >= pt1x) {
                        Pt2y = delim.y - 1;
                  } senão
                  Se (delim.y == atual.y && delim.x >= pt1x) {
                        Pt1x = delim.x + 1;
                  }
            }
      }
      Retangulos.add(RetanguloLivre(pt1x, pt1y, pt2x, pt2y))
}
```

Resultado da aplicação deste algoritmo em nosso pequeno exemplo:

Iteração 1 de delim:



Iteração 2 de delim:

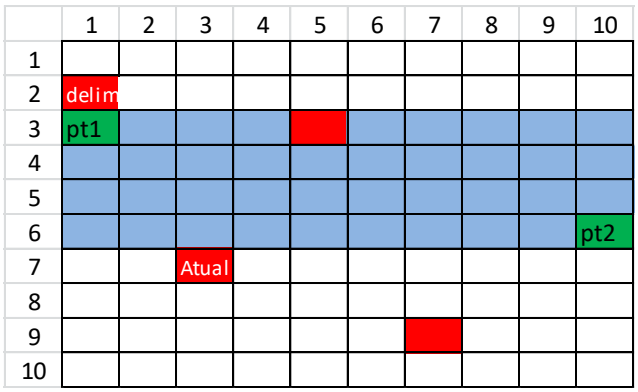


- Delimitação 2
  A segunda forma de delimitar um retângulo na parte superior da mina terrestre atual segue o mesmo padrão da delimitação anterior, porém com algumas mudanças na comparação das 3 condicionais internas:

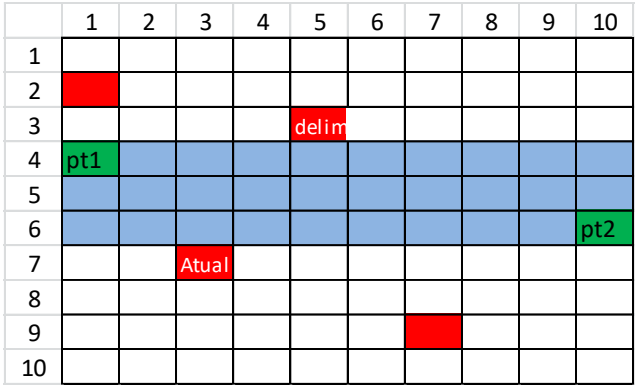
```
Se (delim.x != atual.x -1 && delim.x >= pt1x && delim.x >= pt1x) {
      Pt1x = delim.x + 1;
} senão
Se (delim.y < atual.y && delim.y >= pt1y) {
      Pt1y = delim.y + 1;
} senão
Se (delim.y > atual.y && delim.y <= pt2y) {
      Pt2y = delim.y - 1;
}
```

Resultando em:

Iteração 1 de delim:



Iteração 2 de delim:



Note que os dois retângulos obtidos são diferentes, apesar disso nem sempre ocorrer. O exemplo mostrado se refere aos dois retângulos apenas da parte superior da mina terrestre em questão, e portanto, devemos aplicar o mesmo processo à esquerda, direita, e abaixo da mesma. Dessa forma iremos obter até 8 retângulos para cada mina terrestre que analisarmos.

Para os outros lados, o algoritmo segue o mesmo padrão do que já foi apresentado, alterando-se apenas os valores de inicialização de *pt1x*, *pt1y*, *pt2x*, *pt2y*, e a inversão de alguns operadores lógicos e posicionamento de variáveis durante as comparações internas, e portanto, serão omitidos deste artigo para evitar exposição massiva de pseudocódigo sem relevância adicional significativa ao entendimento da solução. Logo, é importante apenas o entendimento do algoritmo apresentado, simplesmente replicando seu funcionamento aos outros lados.

Ao realizar esse processo para cada mina terrestre presente no campo, temos todos os retângulos livres presentes no campo, armazenados em nossa lista de retângulos. De acordo com os atributos que delimitam um retângulo, podemos realizar o cálculo de sua área, como já apresentado anteriormente, por meio da fórmula: *(pt2x-pt1x+1) * (pt2y-pt1y+1)*. Logo, precisamos iterar sobre a lista de retângulos em busca daquele que possui a maior área.

Considerando que para cada mina terrestre que iteramos, nós iniciamos uma nova iteração sobre a lista, este algoritmo apresenta complexidade $\mathbf{O(n^2)}$, sendo n a quantidade de minas no campo, o que pode ser julgado eficiente ao considerarmos o fato de haver um número relativamente reduzido de células minadas no campo.

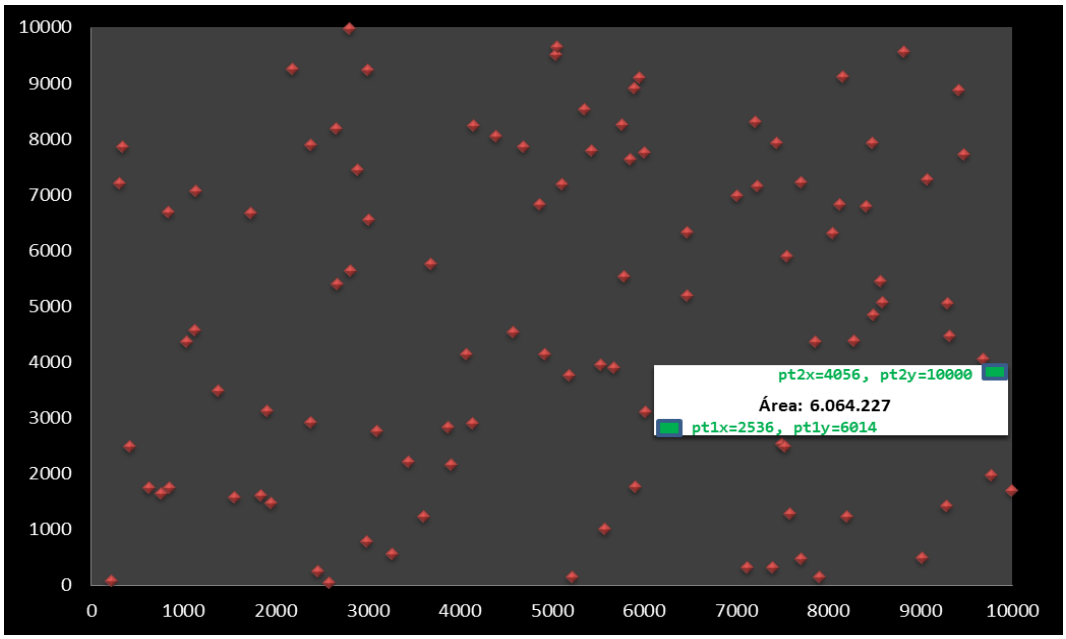
Casos de teste

Para a verificação da efetividade do algoritmo, foram disponibilizados casos de teste, com os seguintes dados:

Nome	Largura	Altura	Minas	Nº de Células	Minas / Células livres
caso010	10.000	10.000	100	100.000.000	1 / 999.999
caso020	20.000	20.000	200	400.000.000	1 / 1.999.999
caso030	30.000	30.000	300	900.000.000	1 / 2.999.999
caso040	40.000	40.000	400	1.600.000.000	1 / 3.999.999
caso050	50.000	50.000	500	2.500.000.000	1 / 4.999.999
caso060	60.000	60.000	600	3.600.000.000	1 / 5.999.999
caso070	70.000	70.000	700	4.900.000.000	1 / 6. 999.999
caso080	80.000	80.000	800	6.400.000.000	1 / 7.999.999
caso090	90.000	90.000	900	8.100.000.000	1 / 8.999.999
caso100	100.000	100.000	1.000	10.000.000.000	1 / 9.999.999

Caso010

Para a exibição dos resultados obtidos, iremos utilizar um gráfico meramente ilustrativo no primeiro caso, onde o ponto esquerdo (pt1x, pt1y) ficará na parte inferior, e o ponto direito (pt2x, pt2y) na parte superior do retângulo encontrado:



Média de tempo gasto: 400 ms

Caso020

Posição do retângulo:

pt1x = 8.369 pt1y = 6.796

pt2x = 9.404 pt2y = 20.000

Área = 13.680.380

Média de tempo gasto: 406 ms

Caso030

Posição do retângulo:

pt1x = 1 pt1y = 4.031

pt2x = 6.627 pt2y = 8.767

Área = 31.392.099

Média de tempo gasto: 1125 ms

Caso040

Posição do retângulo:

pt1x = 1 pt1y = 14.727

pt2x = 18.066 pt2y = 16.527

Área = 32.536.866

Média de tempo gasto: 1453 ms

Caso050

Posição do retângulo:

pt1x = 1 pt1y = 16.626

pt2x = 38.902 pt2y = 17.867

Área = 48.316.284

Média de tempo gasto: 1625 ms

Caso060

Posição do retângulo:

pt1x = 3.767 pt1y = 46.327

pt2x = 60.000 pt2y = 47.456

Área = 63.544.420

Média de tempo gasto: 1860 ms

Caso070

Posição do retângulo:

pt1x = 1 pt1y = 16.788

pt2x = 65.437 pt2y = 18.019

Área = 80.618.384

Média de tempo gasto: 1437 ms

Caso080

Posição do retângulo:

pt1x = 60.673 pt1y = 27.163

pt2x = 62.373 pt2y = 80.000

Área = 89.877.438

Média de tempo gasto: 1547 ms

Caso090

Posição do retângulo:

pt1x = 1 pt1y = 79.169

pt2x = 35.394 pt2y = 81.587

Área = 85.618.086

Média de tempo gasto: 1.563 ms

Caso100

Posição do retângulo:

pt1x = 1 pt1y = 51.915

pt2x = 62.584 pt2y = 53.811

Área = 118.721.848

Média de tempo gasto: 1781 ms

Conclusão

Finalizamos concluindo que esta solução é aceitável para os casos de teste analisados, considerando seu tempo útil de execução. Porém, este algoritmo poderia ainda ser otimizado tanto em termos de memória, ao não inserirmos retângulos repetidos em nossa lista, quanto em termos de processamento, evitando percorrer outras minas terrestres que já estejam fora da área inicialmente delimitada em cada lado da mina terrestre sendo analisada. Como o processo de busca por retângulos acontece conceitualmente igual nos quatro lados de cada mina terrestre, no algoritmo, os parâmetros de comparação entre os valores mudam minimamente, e portanto, muito código similar foi produzido na implementação Java (o que justifica a exibição não completa do código neste artigo), porém, um fator aceitável considerando a clareza proporcionada e o objetivo cumprido com bom desempenho.

Referências

Alguns poucos detalhes inspirados da solução comentada em sala de aula ao desafio disponível em: *https://code.google.com/codejam/contest/3264486/dashboard#s=p2*

Repositório contendo código implementado em Java, destinado à elaboração deste artigo: *https://github.com/CarlosAsrc/T1ALEST2*