Merge Sort Count para resolver o problema do Cortador de Pizza

Pedro Olímpio N. de O. Pinheiro[1], **Wladimir Araujo Tavares**[1]

[1]Universidade Federal do Ceará - Campus Quixadá
Av. José de Freitas Queiroz, 5003 – Cedro – Quixadá – Ceará 63902-580

pedro.olimpio@hotmail.com, wladimirufc@gmail.com

Abstract. *In this paper, we will describe issue C of the regional phase of the XXIII SBC Programming Marathon, classification competition for the ACM International Collegiate Programming Contest, and an efficient way of solving it.*

Resumo. *Nesse artigo será descrita a questão C da fase regional da XXIII Maratona de Programação da SBC, competição classificatória para o ACM International Collegiate Programming Contest, e uma maneira eficiente de resolvê-la.*

1. Introdução

A Maratona de Programação é um evento da Sociedade Brasileira de Computação que existe desde o ano de 1996. A Maratona nasceu das competições regionais classificatórias para as finais mundiais do concurso de programação da ACM, o ACM International Collegiate Programming Contest, e é parte da regional sulamericana do concurso. Ela se destina a alunos e alunas de cursos de graduação e início de pós-graduação na área de Computação e afins (Ciência da Computação, Engenharia de Computação, Sistemas de Informação, Matemática, etc). A competição promove nos estudantes a criatividade, a capacidade de trabalho em equipe, a busca de novas soluções de software e a habilidade de resolver problemas sob pressão. De ano para ano temos observado que as instituições e principalmente as grandes empresas da área têm valorizado os alunos que participam da Maratona.

A fase regional da XXIII Maratona de Programação, realizada em 2018, continha 13 questões, Cortador de Pizza foi uma delas. Para uma solução ser aceita pelos juízes da competição, além de responder corretamente para um conjunto de casos de testes, é necessário obter as respostas dentro de um tempo limite de execução, limitando a complexidade do algoritmo.

2. Cortador de Pizza

A prova da Maratona de Programação de 2018 tinha 13 questões, entre elas a questão Cortador de Pizza (letra C):

Vô Giuseppe ganhou de presente um cortador profissional de pizza, daqueles do tipo carretilha e, para comemorar, assou uma pizza retangular gigante para seus netos! Ele sempre dividiu suas pizzas em pedaços fazendo cortes ao longo de linhas contínuas, não necessariamente retilíneas, de dois tipos: algumas começam na borda esquerda da pizza, seguem monotonicamente para a direita e terminam na borda direita; outras começam na

borda inferior, seguem monotonicamente para cima e terminam na borda superior. Mas Vô Giuseppe sempre seguia uma propriedade: dois cortes do mesmo tipo nunca podiam se interceptar. Veja um exemplo com 4 cortes, dois de cada tipo, na parte esquerda da figura, que dividem a pizza em 9 pedaços.

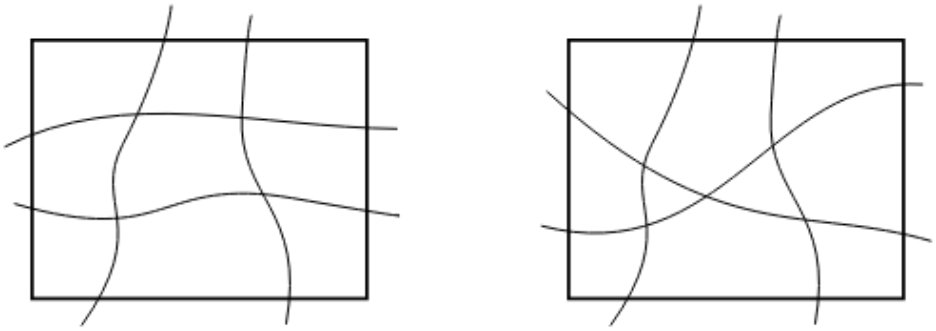

Figura 1. Exemplo de cortes.

Acontece que Vô Giuseppe simplesmente ama geometria, topologia, combinatória e coisas assim; por isso, resolveu mostrar para as crianças que poderia obter mais pedaços, com o mesmo número de cortes, se cruzamentos de cortes de mesmo tipo fossem permitidos. A parte direita da figura mostra, por exemplo, que se os dois cortes do tipo dos que vão da esquerda para a direita puderem se interceptar, a pizza será dividida em 10 pedaços. Vô Giuseppe descartou a propriedade, mas não vai fazer cortes aleatórios. Além de serem de um dos dois tipos, eles vão obedecer às seguintes restrições:

- Dois cortes têm no máximo um ponto de interseção e, se tiverem, é porque os cortes se cruzam naquele ponto;
- Três cortes não se interceptam num mesmo ponto;
- Dois cortes não se interceptam na borda da pizza;
- Um corte não intercepta um canto da pizza.

Dados os pontos de começo e término de cada corte, seu programa deve computar o número de pedaços resultantes dos cortes do Vô Giuseppe.

Entrada

A primeira linha da entrada contém dois inteiros X e Y, $(1 \leq X, Y \leq 10^9)$ representando as coordenadas (X, Y) do canto superior direito da pizza. O canto inferior esquerdo tem sempre as coordenadas $(0, 0)$. A segunda linha contém dois inteiros H e V, $(1 \leq H, V \leq 10^5)$, indicando, respectivamente, o número de cortes que vão da esquerda para a direita, e o número de cortes que vão de baixo para cima. Cada uma das H linhas seguintes contém dois inteiros Y_1 e Y_2 definindo as ordenadas de encontro dos lados verticais da pizza com um corte que vai do lado esquerdo, na ordenada Y_1, para o lado direito, na ordenada Y_2. Cada uma das V linhas seguintes contém dois inteiros X_1 e X_2 definindo as abscissas de encontro dos lados horizontais da pizza com um corte que vai do lado inferior, na abscissa X_1, para o lado superior, na abscissa X_2.

Saída

Imprima uma linha contendo um inteiro representando o número de pedaços resultantes.

Exemplo de entrada 1
3 4
3 2
1 2
2 1
3 3
1 1
2 2

Exemplo de saída 1
13

Exemplo de entrada 2
5 5
3 3
2 1
3 2
1 3
3 4
4 3
2 2

Exemplo de saída 2
19

Exemplo de entrada 3
10000 10000
1 2
321 3455
10 2347
543 8765

Exemplo de saída 3
6

Segue os cortes descritos nos 3 exemplo:

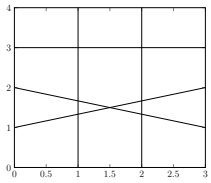

Figura 2. Exemplo 1.

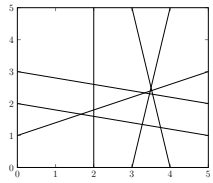

Figura 3. Exemplo 2.

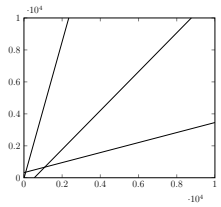

Figura 4. Exemplo 3.

3. Metodologia

A primeira restrição de corte (Dois cortes têm no máximo um ponto de interseção e, se tiverem, é porque os cortes se cruzam naquele ponto) permite considerar todos os cortes como retas, pois se cruzam apenas uma vez.

Para o caso mais simples do problema, onde os únicos cruzamentos de são entre cortes verticais e horizontais, a solução pode ser obtida multiplicando o número de pedaços em cada linha $(V + 1)$ pelo número de pedaços em cada coluna $(H + 1)$. No exemplo abaixo H vale 2 e V vale 1. A quantidade de pedaços é $(H + 1) \times (V + 1) = 3 \times 2 = 6$.

Exemplo de Entrada
6 6
2 1
2 2
4 4
3 3

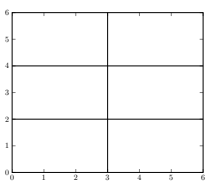

Figura 5. Cortes do exemplo.

As restrições 2 à 4 dos cortes garantem que toda vez que dois cortes se cruzam, um pedaço é adicionado. Como no exemplo abaixo, que possui valores iguais para H e V porém quantidade diferente de pedaços (7), pois existe um cruzamento.

Exemplo de Entrada
6 6
2 1
2 4
4 2
2 2

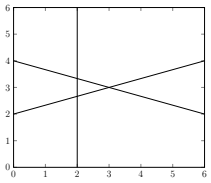

Figura 6. Cortes do exemplo.

Conclui-se que a quantidade final de pedaços é dada pela fórmula:

$$pedacos = (H + 1) \times (V + 1) + cruzamentos \tag{1}$$

O problema foi reduzido a calcular quantos cruzamentos existem entre cortes do mesmo tipo (horizontais ou verticais).

4. Cruzamentos

Dois cortes horizontais se cruzam quando aquele com maior ordenada de início tem uma ordenada de fim menor que a do outro corte. No exemplo abaixo, os dois primeiros cortes não se cruzam (pois o primeiro possui ordenada de início e de fim maior que o segundo), porém, os dois últimos se cruzam (pois o terceiro possui ordenada de início maior do que o quarto, porém ordenada de fim menor):

Exemplo de Entrada
6 6
4 0
5 4
4 3
2 1
1 2

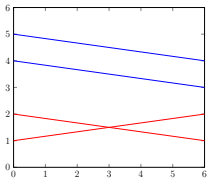

Figura 7. Cortes do exemplo.

Para cruzamentos de cortes verticais funciona de maneira análoga, comparando as abscissas ao invés das ordenadas.

Comparar cada corte com todos os outros, verificando cruzamentos, pode ser demorado, com complexidade $O(n^2)$. Para contar os cruzamentos de maneira mais eficiente, o vetor com os cortes é ordenado em ordem crescente de início de corte, e depois é contado a quantidade de inversões que são necessárias para que o vetor fique ordenado em ordem crescente de final. É possível adaptar o algoritmo do *merge sort* para além de ordenar, contar quantas inversões foram feitas.

A Pseudocódigo 1 mostra o algoritmo do *merge sort* adaptado para contar inversões. Primeiro, usando recursividade, é calculada quantas inversões são necessárias para ordenar a primeira e a segunda metade do vetor separadamente. Os vetores $u1$ e $u2$ são criados, ambos já estão ordenados e é inserido infinito no final de cada um, de maneira que quando um vetor acaba, todos os elementos do outro são inseridos no resultado. A

seguir os vetores $u1$ e $u2$ são percorridos pelas variáveis $ini1$ e $ini2$. Quando o elementos menor está em $u1$ ele apenas é copiado para o vetor original V e avança $ini1$. Caso contrário, o elemento de $u2$ é copiado para V e é incrementado em inv a quantidade de inversões necessárias para leva-lo para essa posição, considerando que $u2$ inicia depois que $u1$ termina, que é o tamanho de $u1(metade)$ menos a quantidade de elementos de $u1$ já inseridos($ini1$).

Algorithm 1 Merge Sort Count

```
function MERGE COUNT(V, inicio, fim)
    n ← fim − inicio
    if n = 1 then
        return 0
    end if
    meio = inicio + (n/2)
    inv = 0
    inv ← inv + Merge Count(V, inicio, meio)
    inv ← inv + Merge Count(V, meio, fim)
    u1 = V[inicio, meio] + {∞}
    u2 = V[meio, fim] + {∞}
    ini1 ← 0
    ini2 ← 0
    for i ← inicio to fim − 1 do
        if u1[ini1] ≤ u2[ini2] then
            V[i] ← u1[ini1]
            ini1 ← ini1 + 1
        else
            V[i] ← u2[ini2]
            ini2 ← ini2 + 1
            inv ← inv + (metade − ini1)
        end if
    end for
    return inv
end function
```

O número de cruzamentos é obtido somando o número de inversões para ordenar o vetor de cortes horizontais e verticais.

5. Conclusão e Resultados

Utilizando esse algoritmo são feitas 4 ordenações, cada uma em $O(n \log n)$, obtendo uma complexidade final $O(H \log H + V \log V)$. Para esse problema, que $H, V \leq 10^5$, essa complexidade é suficiente para um tempo limite de 1 segundo.

Esse problema pode ser encontrado no *URI Online Judge*, número 2878. A submissão com esse algoritmo foi aceita e ficou em segundo lugar no *ranking*, empatado com o primeiro no tempo de execução (0.092 segundos).

Referências

2878 - cortador de pizza - uri online judge. `https://www.urionlinejudge.com.br/judge/pt/problems/view/2878`. Accessed: 2018-10-01.

Xxiii maratona de programação. `http://maratona.ime.usp.br/`. Accessed: 2018-10-01.

Halim, S. and Halim, F. (2013). *Competitive Programming 3*. Lulu Independent Publish.