A solução adotada para resolver a separação das barracas foi a formação de polígonos a partir das coordenadas das barracas e verificar se os polígonos gerados a partir do invólucro das barracas se sobrepõe.

A 1ª parte do código pega as entradas de um arquivo informado em “open()”, as converte em números inteiros e as organiza para formar os pontos do contorno das barracas.

A 2ª parte do código cria vetores para conter todos os vértices de cada grupo de barracas, depois adiciona os vértices dos polígonos de cada grupo no vetor com um “for” para todas entradas do grupo. E por fim, salva um novo polígono com todos os vértices do vetor e então reduz o polígono para o seu próprio invólucro com a propriedade “convex_hull” (“invólucro convexo”).

E a última parte do código é uma comparação utilizando duas funções internas da classe de polígonos, as funções “contains()” e “overlaps()”, que verificam respectivamente se um polígono contém o outro em sua área, e se um polígono contém somente uma parte de sua área sobre o outro. Caso um dos resultados verificados for verdadeiro é apontado que não é possível separar as figuras, e caso contrário, é apontado que é possível separar.

A parte mais complexa do algoritmo é a parte da formação dos invólucros que é feita por internamente na biblioteca Shapely da linguagem Python, usada na formação dos polígonos e suas operações. Como a implementação não é nada tão complexa e específica com o uso de polígonos, o interpretador da linguagem opta por usar o algoritmo de invólucro convexo de Graham Scan.

O algoritmo é um algoritmo para traçar invólucros convexos de complexidade O(nLogn), que consiste em pegar o ponto mais baixo da figura, analisar os pontos com menor ângulo (e distância, caso o ângulo seja o mesmo), e ligar os pontos de forma que o polígono resultante seja a forma da “periferia” dos pontos, ou seja o invólucro.

Analisando a complexidade da solução das barracas, temos que para cada entrada com n1 barracas masculinas e n2 barracas femininas uma função “13 + n1*6 + n2*6 + n1 + n1*log(n1) + n2 + n2*log(n2)” em que o número independente (13) é a soma das funções individuais de abertura de arquivo, inicialização de variáveis, condicionais e operações matemáticas que não pertencem a um loop “for”, as multiplicações “n1*6” e “n2*6” são os loops para leitura dos pontos de entrada, os números “n1” e ”n2” são os loops dos vetores de vértices e as multiplicações “n1*log(n1)” e “n2*log(n2)” são as formações dos invólucros convexos dos polígonos. A expressão resultante para esse algoritmo será “n1*(7 + log(n1)) + n2*(7 + log(n2)) + 13”, porém como no pior caso do algoritmo n1 = n2 = n, a expressão resultante será “2*n*(7 + log(n))”, cuja representação está abaixo. Além disso, como o maior item da expressão é “2*n*log(n)”, podemos afirmar que a expressão é contida em O(n*log(n)).

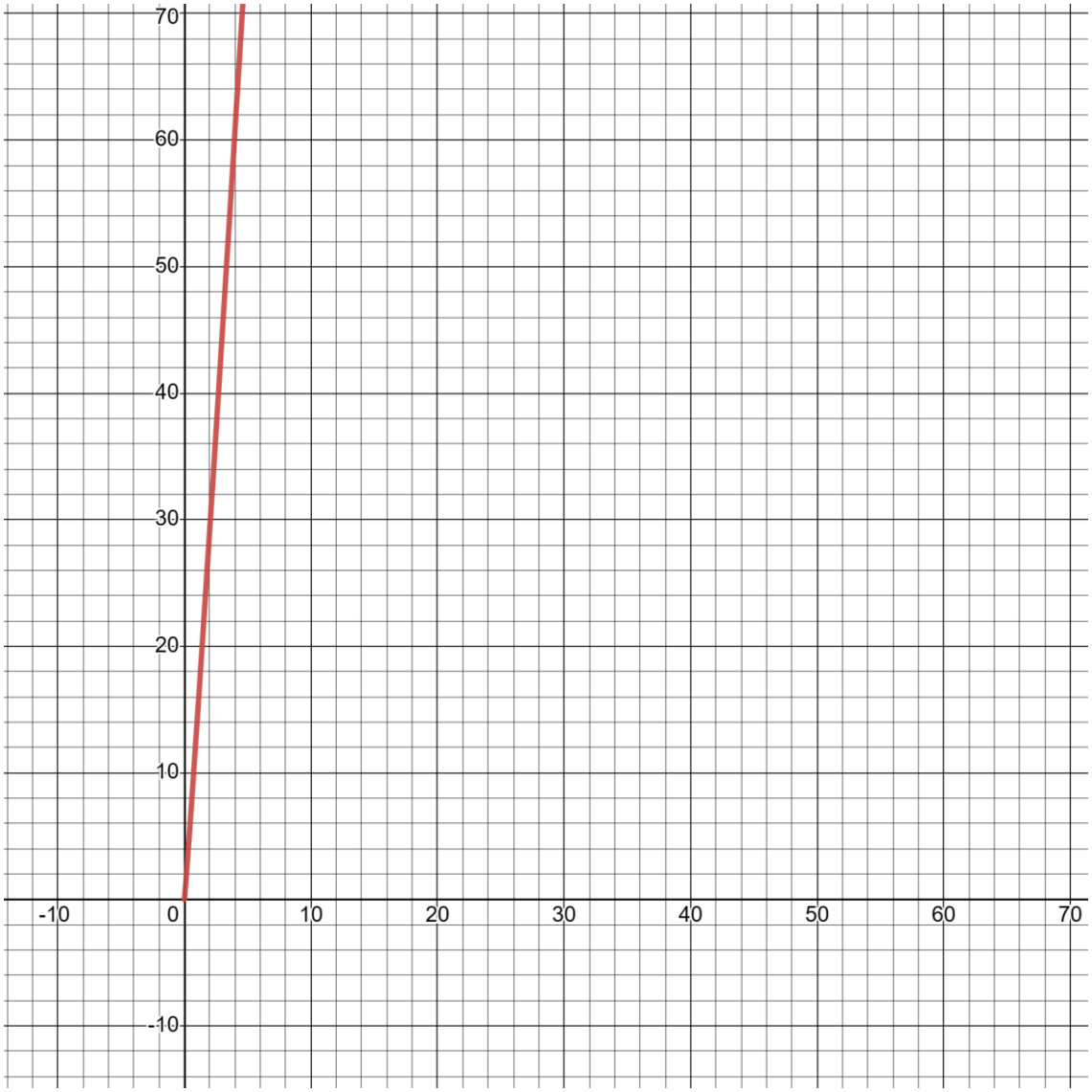

Gráfico da expressão final resultante – Feito com a ferramenta online Desmos – Disponível em: <https://www.desmos.com/calculator/zwt6r9kpo4>