**Trabalho Prática 2**

**Fecho Convexo**

**Eduardo Silveira Cezar Fernandes – 2021019424**

Departamento de Ciência da Computação – Universidade Federal de Minas Gerais(UFMG)

Belo Horizonte – MG – Brazil

[eduardoscf1210@gmail.com](mailto:eduardoscf1210@gmail.com)

**1. Introdução**

Esta documentação lida com o problema de determinar qual é o menor fecho convexo possível dado um conjunto de pontos cartesianos. Para concretizar a funcionalidade proposta, o programa pode ser utilizado com 2 métodos principais, são eles: Scan de Graham e Marchar de Jarvis. Sendo que o primeiro deles pode ser executado com 3 métodos de ordenação diferente, o MergeSort, o InsertionSort e o BucketSort.

**2. Método**

**2.1. Configurações da máquina**

Sistema operacional:

Linguagem de programação: C++

Compilador: G++ / GNU Compiler Collection

Processador: Intel(R) Core(TM) i7-5500U CPU @ 2.40GHz 2.40 GHz

Memória RAM: 12,0 GB

**2.2. Estruturas de dados**

A estrutura de dados utilizada nesse trabalho foi o array. O uso dessa estrutura específica teve em vista armazenar e lidar com os pontos que o programa irá receber.

Há um array de pontos cartesianos alocado dinamicamente que irá receber todos os pontos do arquivo lido, pois dessa forma o programa será capaz de lidar com qualquer quantidade de pontos que o arquivo fornecer. Com este array, todas as operações do programa serão executadas a partir dele.

Já no Scan de Graham e no Marchar de Jarvis, foram criados array temporários dos pontos que pertencerão ao fecho convexo para então criar as retas formadas por esses pontos.

**2.3. Classes**

Na implementação desse trabalho, foram utilizadas 3 classes, PontoCartesiano, reta e FechoConvexo.

Como o programa recebe pontos cartesianos, é necessário ter um tipo para armazená-los, já na hora que o fecho convexo está sendo criado, primeiro são criadas as retas que vão compor o fecho convexo e em seguida essas retas são adicionadas.

A classe PontoCartesiano possui a maioria das funções e os métodos funcionam ao redor dela, ou seja, a ordenação é feita em um array de pontos.

A classe Reta possui os dois pontos que a formam e irá compor o fecho convexo.

A classe FechoConvexo irá ser formada a partir de retas e é objetivo principal do programa, ter esse fecho formado corretamente.

**2.4. Funções**

O código possui 5 funções principais, a mergeSort(), a insertionSort(), a bucketSort, o scanDeGraham(), e por fim o marcharDeJarvis().

As funções mergeSort, insertionSort e bucketSort possuem o mesmo objetivo: ordenar o array de pontos lido em relação ao que o Scan de Graham pede. Ou seja, define o primeiro como aquele mais embaixo (menor coordenada Y) e em caso de mais de um ponto com a mesma coordenada Y, o desempate é feito com o menor X. Após isso, todos os demais pontos são ordenados de forma crescente em relação ao ângulo formado com esse ponto inicial. OBS: O ponto inicial é encontrado por uma função separada e este ponto é passado como parâmetro para esses métodos de ordenação.

A função Scan de Graham recebe o array já ordenado, com isso o algoritmo procede considerando cada um dos pontos do array ordenado em sequência. Para cada ponto, é determinado, se ao mover-se dos dois pontos anteriores para este ponto se forma uma "curva no sentido anti-horário" ou uma "curva no sentido horário". Se é uma "curva para esquerda", isto significa que o ponto de partida não faz parte do envoltório convexo e deve ser removido da pesquisa. Este processo continua ao longo do conjunto até que o conjunto dos três últimos pontos seja uma curva para direita. Assim que uma "curva no sentido anti-horário" é encontrada, o algoritmo salta para o próximo ponto do array ordenado.

Já o Marchar de Jarvis não precisa de ordenação. O algoritmo começa pelo ponto mais a esquerda, ou seja, aquele que possui menor coordenada X, depois continuamos percorrendo os pontos em sentido anti-horário. O ponto é atribuído ao fecho convexo se ele possui a maior orientação no sentido anti-horário em relação aos demais.

Outras funções como orientacao(), calcularAngulo() e encontrarPrimeiroPonto são funções auxiliares para serem utilizadas por esses métodos principais.