

## Projeto I : Fecho Convexo

Data de Entrega: 30/10/2020

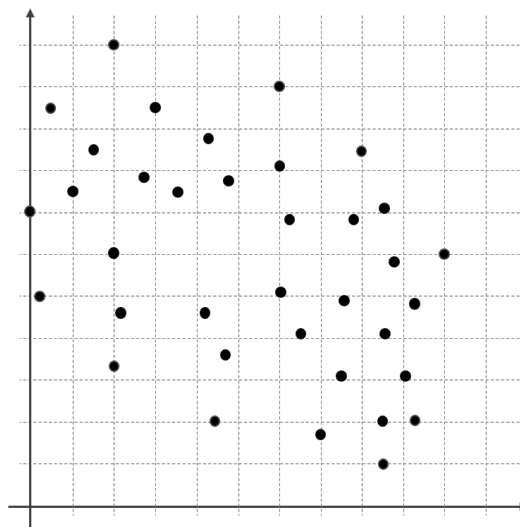
Este projeto deverá ser feito em **duplas**. A entrega deverá ser realizada via [run.codes](#), fazendo *upload* dos arquivos em “**PROJETO I 30/10**”.

### Descrição do Problema

O fecho convexo é um importante problema geométrico definido da seguinte maneira:

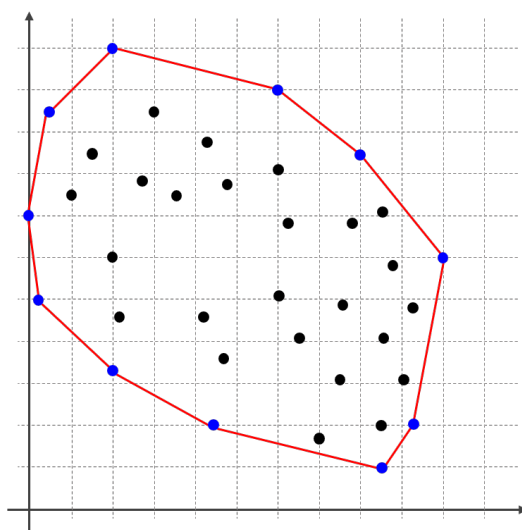
*“Dado um conjunto de pontos  $S$ , o fecho convexo é o polígono convexo de menor área que contém todos os pontos em  $S$ .”*

Para visualizar, imagine que o conjunto de pontos  $S$  são pregos numa tábua. O fecho convexo é o formato que um elástico tomará após ser esticado para englobar todos os pregos, tocando naqueles mais afastados do centro.



**Figura 1.** Exemplo de conjunto com 36 pontos

A Figura 1 ilustra um conjunto em um espaço bidimensional com  $n=36$  pontos. O fecho convexo para esse conjunto de pontos está ilustrado na Figura 2 e possui  $h=11$  ( $h$  é a quantidade de pontos do polígono convexo). Os pontos em azul são aqueles que pertencem ao conjunto que compõem o fecho convexo.



**Figura 2.** Fecho convexo do conjunto

A determinação do fecho convexo possui inúmeras aplicações relevantes, como identificação e classificação de objetos, medida de similaridade de forma, busca de caminhos, otimização inteira, entre outras.

## Objetivo do Trabalho

O trabalho tem como principal objetivo desenvolver um programa para **encontrar o fecho convexo** para um conjunto  $S$  com  $n$  pontos, ou seja, encontrar os  $h$  pontos contidos em  $S$  que compõem o polígono convexo de menor área que engloba todos os pontos de  $S$ .

O desenvolvimento do trabalho envolve as seguintes atividades principais:

1. Modelar a solução para o cálculo do fecho convexo utilizando **duas Estruturas de Dados diferentes** vistas em aula (até a Aula 11).
2. Desenvolver um algoritmo que encontre o fecho convexo.
3. Implementar a solução em Linguagem C, usando (obrigatoriamente) o conceito de **TAD**, modularização de código e Makefile para compilação e geração da aplicação executável.
4. Apresentar um gráfico de crescimento de tempo de execução para diferentes tamanhos de entrada.

Essas atividades compõem as 3 partes do trabalho, descritas a seguir.

## Parte I – Modelagem da Solução (25% da nota final)

Serão avaliadas a coerência e relevância da solução e respectiva justificativa para os seguintes aspectos:

- Quais Estruturas de Dados e respectivas abordagens de implementação foram escolhidas para modelar o problema? Justifique.
- Quais vantagens e limitações as estruturas/implementações escolhidas apresentam?
- Explique claramente e detalhadamente a lógica utilizada para resolver o problema. Como as estruturas (e respectivas operações básicas) são utilizadas para representar o problema? Qual o algoritmo utilizado para resolver o problema?

## Parte II – Implementação (45% da nota final)

Esta parte avalia a estratégia de implementação utilizada. Por isso, serão considerados aspectos como organização e corretude do código, e documentação interna.

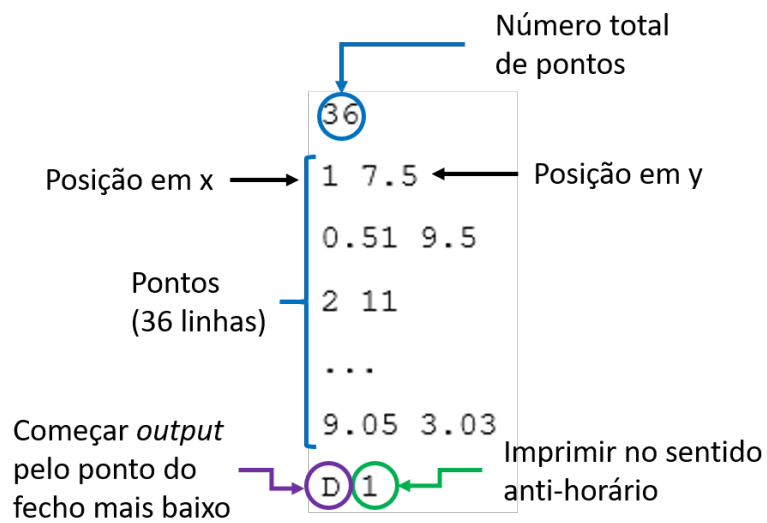
A manipulação das Estruturas de Dados deve ser baseada em **TAD** e o código deve ser escrito em **Linguagem C** (C99, flag -Wall), compilado com **gcc**. Além disso, sua implementação deve atender os requisitos descritos a seguir.

**Entrada do Programa:** Serão disponibilizados 3 arquivos de teste que mostram como deverá ser a entrada do programa. A primeira linha dos arquivos indica a quantidade de pontos ***n*** que compõem o conjunto e as ***n*** linhas subsequentes contêm as **coordenadas dos pontos** em  $(x, y)$ . A última linha indicará o **formato de saída**, contendo duas informações:

1. Ponto pelo qual deverá ser iniciada a impressão do polígono
  - **L**: ponto mais à esquerda;
  - **R** ponto mais à direita;
  - **D** ponto mais baixo;
  - **U** ponto mais alto.
2. Sentido a ser seguido para impressão do polígono
  - **0** para sentido horário,
  - **1** para sentido anti-horário.

O número de total de pontos será um inteiro variando de **3** a **1.000.000**, incluso.

### Exemplo da Entrada



O programa deve ler um conjunto de entrada, armazenar os dados e calcular o fecho convexo.

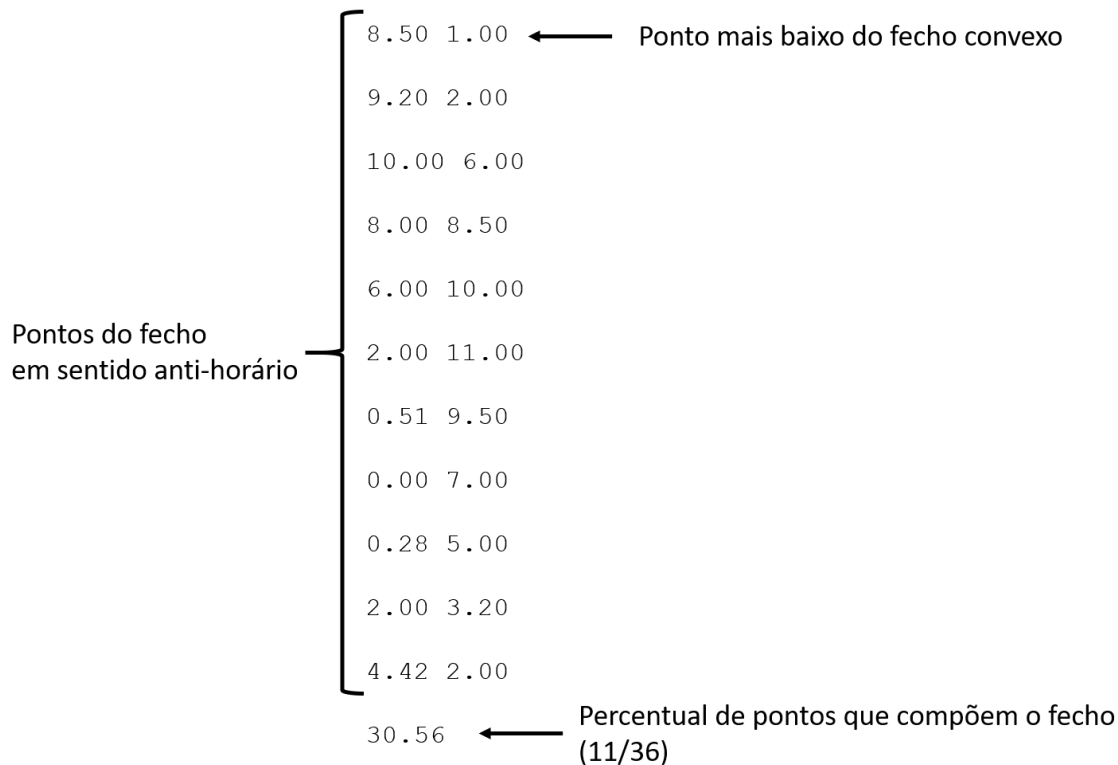
**Saída do Programa:** O programa deve apresentar como saída:

1. o **maior conjunto** de pontos que compõem o fecho convexo (ou seja, os pontos mínimos necessários e os seus colineares), começando pelo ponto e seguindo pelo sentido informados na última linha da entrada;
2. **percentual de pontos** do conjunto que compõem o fecho convexo.

Todas as saídas devem ser dadas com **duas casas decimais** de precisão.

Caso não seja possível formar o fecho convexo, imprima apenas **ERRO** na saída.

### Exemplo de Saída



Considere que, se:

- **L** e existe mais de 1 ponto mais à esquerda, pegar o mais à esquerda mais inferior;
- **R** e existe mais de 1 ponto mais à direita, pegar o mais à direita mais inferior;
- **D** e existe mais de 1 ponto mais baixo, pegar o mais baixo mais à esquerda;
- **U** e existe mais de 1 ponto mais alto, pegar o mais alto mais à esquerda.

Vale ressaltar que os arquivos de teste fornecidos têm como objetivo apenas apresentar o formato esperado para a entrada de dados e direcionar os testes iniciais. Na correção, seu programa será testado nos conjuntos disponibilizados e em conjuntos diferentes. Portanto, teste seu programa em conjuntos elaborados pelo próprio grupo.

### Parte III – Análise de Eficiência e Relatório (30% da nota final)

A principal atividade desta parte é avaliar a eficiência (tempo de execução) do algoritmo implementado por meio de [análise de algoritmo](#) (*Big O*) e [análise experimental](#), com gráfico de escalabilidade. Para tanto:

1. Meça o tempo de execução para processamento de, pelo menos, os conjuntos de entrada fornecidos para esse fim (arquivo [teste\\_grafico.zip](#)). Use a função `seconds()` do TAD `tempo.h` disponibilizado no Tidia.

Contabilize apenas o tempo para encontrar o fecho convexo, ou seja, **desconsidere tempos de entrada/saída**. Para as execuções:

- a. feche qualquer software desnecessário que esteja rodando no computador, a fim de evitar “interferências”;
- b. execute pelo menos 5 vezes cada conjunto de dados e use a média do tempos contabilizados.

Exemplo de como medir o tempo

Tempo do sistema antes de  
executar o trecho a ser medido



```
double start = seconds();  
/*  
 * trecho do código a ser medido o tempo  
 */  
double end = seconds() - start;
```



Tempo decorrido na  
execução do trecho acima

2. Elabore um gráfico de escalabilidade, que apresenta as medidas de tempo de execução do cálculo do fecho convexo (sem contabilizar leitura da entrada e saída do código) em função do tamanho do conjunto de entrada.
3. Faça a análise de complexidade do algoritmo implementado usando notação Big O ( $O()$ ). Não é necessário detalhar a função, mas é importante apresentar, em linhas gerais, como “encontrou” a complexidade apresentada.
4. Explique o resultado: relacione o resultado apresentado no gráfico à complexidade computacional (Big O) do algoritmo implementado. Discuta sobre fatores que contribuem para o crescimento do tempo de execução, e melhor e pior casos.

## Entrega do Trabalho

Faça um **relatório** de no máximo **5 páginas** com todas as informações e requisitos apresentados nas Partes I, II e III. Sobre a Parte II, inclua no relatório apenas a informação necessária para compilação e execução do seu programa, indicando quais são os arquivos dos TADs, do programa principal, e demais arquivos criados. Não inclua código no relatório.

O arquivo **.zip** contendo o **relatório** (em **pdf**), o **Makefile** e **todo código fonte** necessários para compilar e executar o programa, incluindo arquivos.c (*main* e outros) e

arquivos.h deve ser enviado via *run.codes* em **Projeto I – 30/10**. Não serão aceitos/considerados arquivos executáveis.

O trabalho poderá ser entregue até **30/10/2020, 23:59h**. Trabalhos submetidos em datas posteriores à exigida terão 1,0 ponto de desconto por dia de atraso. **Apenas um integrante da dupla precisa submeter o trabalho no *run.codes*.**

## Referências

1. Fecho Convexo
  - a. *Envoltória Convexa*. Disponível em:  
[https://pt.wikipedia.org/wiki/Envolt%C3%B3ria\\_convexa](https://pt.wikipedia.org/wiki/Envolt%C3%B3ria_convexa). Acessado em 30 de setembro de 2020.
  - b. *Fecho Convexo 2D*. Disponível em:  
<https://www.ime.usp.br/~freitas/gc/fecho.html>. Acessado em 30 de setembro de 2020.
2. Polígono
  - a. *Polígono Convexo*. Disponível em:  
[https://pt.wikipedia.org/wiki/Pol%C3%ADgono\\_convexo](https://pt.wikipedia.org/wiki/Pol%C3%ADgono_convexo). Acessado em 30 de setembro de 2020.

## Apêndice Leitura de Arquivo de Entrada

Para executar seu código em casos teste muito grandes, pode ser necessário passar diretamente o arquivo de entrada. Para fazer isso, altere a linha **run** do seu Makefile para

```
./executavel <arquivo_de_entrada
```