Relatório do Projeto 1: Fecho convexo

SCC0223 - Estruturas de Dados I

Hiago Américo, nUSP: 11218469. Francisco Dias, nUSP: 4402962.

Objetivo: O cálculo do ***fecho convexo*** é um problema clássico de Geometria Computacional, sendo utilizado em inúmeras aplicações no dia-a-dia. O programa descrito neste relatório recebe como entrada um conjunto de pontos com coordenadas (x, y) e devolve o fecho convexo deste conjunto, através dos algoritmos conhecidos como **Algoritmo de Embrulho** e **Algoritmo de Graham**.

1. Modelagem

Ao longo deste relatório, explicaremos sucintamente os TADs utilizados, sendo também possível encontrar informações específicas de nossa implementação em linguagem C na documentação do código.

Uma de nossas primeiras decisões no escopo da modelagem do projeto foi utilizar *Listas Encadeadas Dinâmicas* para armazenar um conjunto L de pontos , o qual temos como objetivo obter o respectivo fecho convexo, aqui também denominado $Conv(L)$.

Uma das principais vantagens deste TAD é que não precisamos pré-definir um tamanho máximo para a quantidade de elementos, ficando este apenas à quantidade de memória principal disponível, não havendo alocação desnecessária de espaço, sendo de responsabilidade do programador a desalocação posterior dos elementos, para que não ocorram vazamentos de memória.

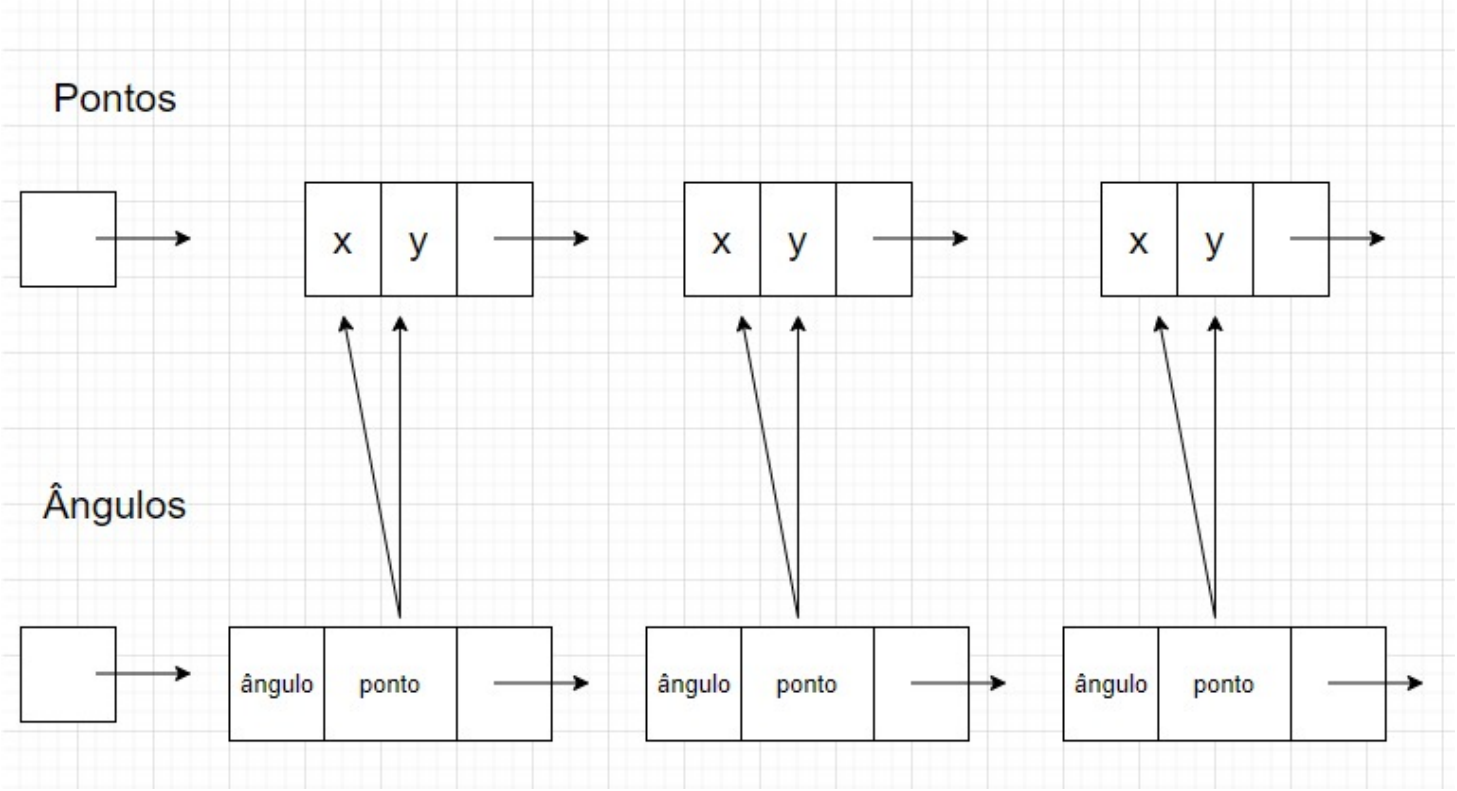
Em contrapartida, temos um maior grau de complexidade exigido para alocações no heap do que a acessos em elementos estáticos na memória. Além disso, não temos o acesso direto como teríamos a elementos armazenados dentro de um vetor, sendo necessário percorrer a lista com o uso de ponteiros auxiliares.

Uma das primeiras abordagens descritas pelos autores da área é conhecida como **Algoritmo do Embrulho**, que é uma solução simples e intuiva para o problema. Abaixo, um esboço em pseudo-código:

```
escolha o ponto com menor coordenada y e inclua-o no fecho
encontre o ponto que possua o menor ângulo entre o primeiro e a reta horizontal
    inclua-o no fecho

para todos os outros pontos de L
  enquanto não incluir o ponto com menor y novamente, faça:
    encontre o ponto que possui maior ângulo e inclua-o no fecho
```

Além da lista ligada, para realizar esta implementação precisaremos de outros TADs auxiliares, para armazenar o ângulo entre os vetores visitados no processo iterativo para poder compará-los e encontrar o máximo ou mínimo deste conjunto, conforme ilustrado na Figura 1.



Embora este seja um método intuitivo para calcular o fecho convexo de um dado conjunto de pontos, ele não o faz da maneira mais eficiente do ponto de vista computacional, como veremos na sessão 3. A segunda implementação proposta para resolver ao problema é conhecida como **Algoritmo de Graham**.

Inicializado de maneira análoga ao algoritmo do embrulho, em sequência esta rotina ordena ao vetor de ângulos formados e vai empilhando os pontos que podem fazer parte de $Conv(L)$, e desempilhando-os caso descubra que são pontos interiores, através de operações de comparação esquerda-direita. Ao final da execução, temos uma pilha de pontos que forma ao fecho convexo.

Como já mencionado, para esta implementação foi necessário utilizar um TAD do tipo *Pilha*. Uma de suas vantagens é a realização de inserções e remoções de elementos em tempo constante, pois essas operações sempre são realizadas no topo.

2. Execução

Abaixo, uma relação das bibliotecas desenvolvidas na implementação do projeto, com uma breve descrição:

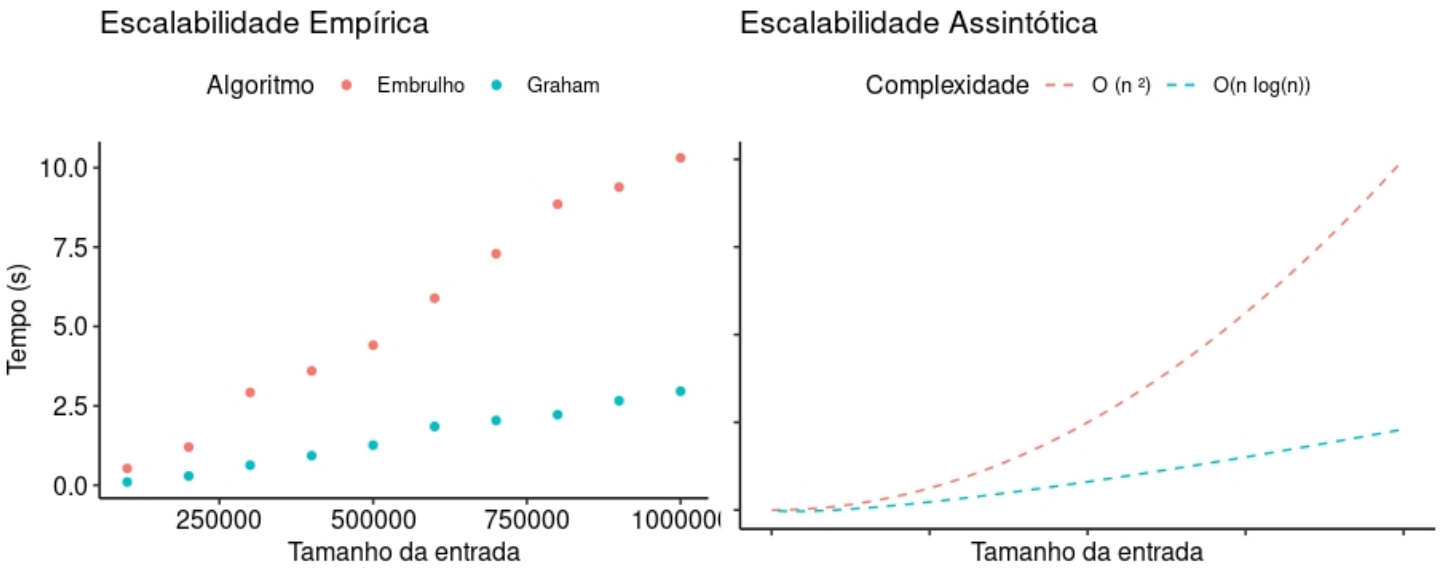
- embrulho.h - contém as funções utilizadas para cálculo do fecho convexo através do **Algoritmo de Embrulho**;
- graham.h - contém as funções utilizadas para cálculo do fecho convexo através do **Algoritmo de Graham**;
- lista.h - contém as funções utilizadas para operações com o TAD *Lista Encadeada Dinâmica*;
- pilha.h - contém as funções utilizadas para operações com o TAD *Pilha*;
- primitivas.h - contém as operações primitivas com vetores para o cálculo do fecho convexo;
- mergesort.h - biblioteca do algoritmo de ordenação MergeSort para listas encadeadas;
- tempo.h - biblioteca para calcular o tempo tomado pelas operações.
- main.c - programa principal, responsável por ler a entrada, chamar às funções e imprimir a saída.

O programa pode ser compilado através da diretiva `make all` e rodado através do comando `make run`. Para limpar os arquivos gerados pela compilação, utilize ao comando `make clean`, conforme padrões vigentes da GNU Coding Standards.

Os padrões de entrada e saída foram gerados conforme especificação do projeto, para serem enviados na plataforma *run.codes*.

3. Análise de Eficiência

- Lista Encadeada Dinâmica
  - $O(1)$: inserção
  - $O(n)$: impressão, exclusão
- Pilha
  - $O(1)$: inserção, remoção
  - $O(n)$: exclusão
- Algoritmo do Embrulho
  - **Operações primitivas**: ângulo entre vetores, máximos e mínimos de um conjunto.
  - **Complexidade**: $O(n^2)$
- Algoritmo de Graham
  - **Operações primitivas**: ângulo entre vetores, máximos e mínimos de um conjunto.
  - **Complexidade**: $O(klog(n))$, sendo k o número de vértices do fecho.



- Medir tempo de execução
- gráfico de escalabilidade
- análise de complexidade utilizando O
- explicar o resultado, relacionando o algoritmo utilizado com a complexidade computacional do algoritmo implementado. Crescimento do tempo, melhor e pior casos.