Implementação de Dois Algoritmos para a Construção de Fecho Convexo Bidimensional.

Rodrigo Nonamor Pereira Mariano de Souza
Eduardo L. Siegfried Blanco
IME-USP

19 de julho de 2000

Sumário

1 Introdução.

Descrevemos neste relatório uma implementação e testes dos algoritmos de Jarvis e Graham para a construção do fecho convexo de um conjunto de pontos no plano.

O fecho convexo de um conjunto de pontos pode ser definido, *grosso modo*, como a fronteira do polígono que contém os pontos do conjunto e cujos vértices são pontos do conjunto. Algoritmos para a construção do fecho convexo fazem parte de uma área de pesquisa conhecida como *Geometria Computacional*, e começaram a ser estudados nos anos 60. Há diversas aplicações desses algoritmos, por exemplo, em robótica.

O algoritmo de Graham é ótimo, ou seja, sua complexidade é igual à cota inferior do problema: $O(n \log n)$. O algoritmo de Jarvis é *output sensitive*, pois sua complexidade depende do número h de arestas do fecho: $O(hn) = O(n^2)$. O algoritmo de Jarvis é também conhecido na literatura como *Embrulho para Presente*, pela maneira como constrói o fecho (para uma visualização da execução, basta imaginar que os pontos são pregos e que o fecho é construído enrolando–se um barbante no conjunto). Não serão feitas descrições dos algoritmos.

2 Implementação.

Os algoritmo foram implementados em C. Segue uma descrição dos fontes:

- **estruturas.h**: Definição de macros e de tipos de objetos geométricos.
- **primitivas.h**: Protótotipos das primitivas Area2 e Left.
- **fecho2.h**: Protótipos das implementações dos algoritmos. Os algoritmos de Graham e Jarvis são implementados, respectivamente, nas funções *graham* e *jarvis*.
- **primitivas.c**: Implementação das primitivas.
- **fecho2.c**: Implementação dos algoritmos.
- **principal.c**: Interface para a chamada dos algoritmos.

O programa é executado com

```
principal [npontos].
```

Quando chamado sem argumentos, pede–se o fornecimento de um arquivo contendo o conjunto de pontos. O formato do arquido de dados é o seguinte:

$\langle$Número de pontos$\rangle$
$p_1.x \quad p_1.y$
$p_2.x \quad p_2.y$
$\vdots$
$p_n.x \quad p_n.y$

Número de pontos é inteiro, e indica o número de pontos do conjunto. Os demais valores representam as coordenadas dos pontos, e podem ser escritos com ponto flutuante.

O argumento opcional faz com que o programa gere um conjunto de pontos de coordenadas aleatórias. O argumento é inteiro positivo, e indica o número de pontos do conjunto a ser gerado.

O programa gera dois arquivos: *graham.out* e *jarvis.out*, que contêm, respectivamente, o fecho encontrado pelos algoritmos de Graham e de Jarvis.

A implementação é original, ou seja, não foi baseada em nenhuma outra implementação existente, e é bastante simples, não incluindo possíveis otimizações.

3 Testes.

Os testes consistiram na execução dos algoritmos para conjuntos de pontos de coordenadas aleatórias. Foram testados conjuntos de tamanhos variados e anotados os tempos de execução. Os tempos foram medidos com o número de pulsos do relógio do processador[1], obtidos com a função *clock* da biblioteca padrão *time.h*. Seguem os resultados obtidos:

N.o de pontos	Pontos no fecho	Tempo Alg. Jarvis (ticks)	Tempo Alg. Graham (ticks)
100	8	30000	20000
1000	18	90000	30000
5000	19	350000	60000
10000	24	840000	110000
13000	24	1120000	130000
15000	27	1420000	150000
18000	24	1510000	190000
20000	28	1950000	190000
23000	27	2180000	240000
25000	28	2460000	250000
28000	26	2540000	280000
30000	27	2850000	300000
31000	27	2940000	320000

Como pode–se perceber, o algoritmo de Graham se mostrou mais eficiente que o de Jarvis em todas as instâncias. Um dado interessante é a verificação de que o número de pontos do fecho cresce muito lentamente com o tamanho da instância.

Na Figura 1 há um gráfico dos resultados. Foram também traçadas duas curvas, correspondentes às funções $c_1 n \log_2 n$ e $c_2 n \log_2 n$, onde c_1 e c_2 são constantes ($c_2 > c_1$). Nota–se que o tempo de execução do algoritmo de Graham acompanha a curva $c_1 n \log_2 n$, confirmando o resultado teórico. O algoritmo de Jarvis parece apresentar um crescimento ligeiramente superior à curva $c_2 n \log_2 n$.

[1] Os autores não sabem que unidade de tempo é essa...

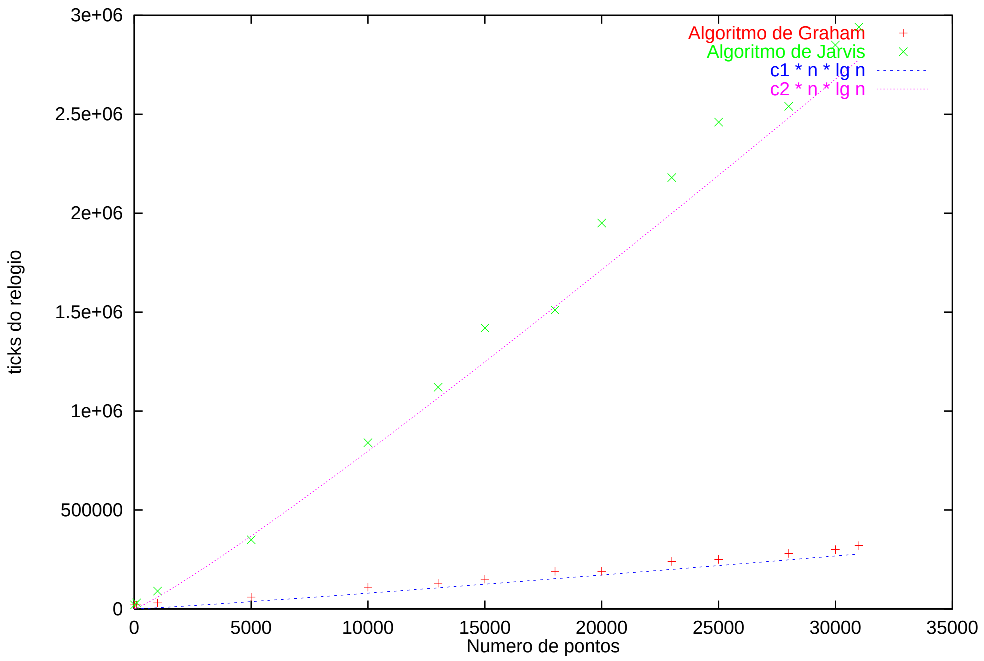

Figura 1: Tempo de Execução dos Algoritmos de Graham e Jarvis.