ICC 2 – Trabalho 03

Prof. Rodrigo Mello

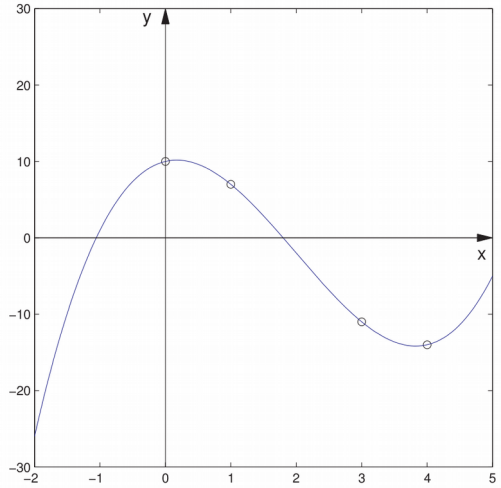
Estagiários PAE

Martha Dais Ferreira
Fausto Guzzo da Costa

Data Máxima para Perguntas: 10/09/2014
Data de Entrega do Trabalho: 15/09/2014
Fórum para Envio de Perguntas: https://groups.google.com/d/forum/icc2-2014
Sistema de Submissões: https://run.codes

Descrição

Considere um conjunto de pontos em um espaço R^2 tal como no gráfico abaixo:



em que foram plotados os pontos:

(0, 10)
(1, 7)
(3, -11)
(4, 14)

O polinômio que intercepta os pontos foi encontrado com o emprego do método Gauss-Jordan visto no trabalho anterior. Por que isso é legal? Dado um conjunto de pontos no espaço, você pode calcular os coeficientes de um polinômio que intercepta esses pontos. Assim você sai de um conjunto discreto de pontos para uma função contínua, que permite estudar o comportamento de seus dados (ou pontos).

Por exemplo, considere que desejamos encontrar os coeficientes para o polinômio abaixo:

$$p(x) = ax^3 + bx^2 + cx + d$$

E que temos o conjunto de pontos em R^2:

(0, 10)
(1, 7)
(3, -11)
(4, -14)

Inicialmente devemos substituir os pontos no polinômio, por exemplo, para o primeiro ponto (0,10), temos:

$p(x) = ax^3 + bx^2 + cx + d$
$10 = a0^3 + b0^2 + c0 + d = d$

Sendo assim, resta o coeficiente d.

Em seguida, aplicamos para o segundo ponto (1,7) e temos:

$p(x) = ax^3 + bx^2 + cx + d$
$7 = a1^3 + b1^2 + c1 + d = a + b + c + d$

Em seguida, aplicamos para o terceiro ponto (3, -11) e temos:

$p(x) = ax^3 + bx^2 + cx + d$
$-11 = a3^3 + b3^2 + c3 + d = 27a + 9b + 3c + d$

Em seguida, aplicamos para o quarto ponto (4, -14) e obtemos:

$p(x) = ax^3 + bx^2 + cx + d$
$-14 = a4^3 + b4^2 + c4 + d = 64a + 16b + 4c + d$

Em seguida, montamos um sistema linear utilizando os resultados da aplicação dos pontos, que tivemos acesso, ao polinômio na forma:

			d	=	10
a	+b	+c	+d	=	7
27a	+9b	+3c	+d	=	-11
64a	+16b	+4c	+d	=	-14

Em seguida, aplica-se o método de Gauss-Jordan e obtém-se os valores do vetor X em que:

$$X = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

Observe que, dessa maneira, pode-se encontrar os coeficientes para qualquer polinômio que desejarmos, desde que tenhamos um conjunto de pontos suficiente para isso.

Seu programa irá receber como entrada o número k de coeficientes do polinômio. Se k for igual a 3, seu polinômio é de grau 2 na forma $p(x) = ax^2 + bx + c$. Se k for igual a 8, seu polinômio será de grau igual a 7 na forma $p(x) = ax^7 + bx^6 + cx^5 + dx^4+ex^3+fx^2+gx+h$. E assim sucessivamente. Em seguida são recebidos k pares de pontos.

Dicas

1) Utilizem precisão de seis casas decimais para impressão;
2) Observem que para resolvermos um sistema linear com k coeficientes, são suficientes k pontos aplicados ao polinômio;
3) Estudem as páginas 56 e 526 do livro https://www.dropbox.com/s/aa71ogpk8xski1j/gaalt1.pdf que contêm esse problema resolvido;
4) Você deve utilizar alocação dinâmica!

Motivação

1) Encontrar polinômios para conjuntos de pontos. Por exemplo, imagine que você fez experimentos em Laboratório de Física e gostaria de encontrar a tendência para um conjunto de pontos. Você pode analisar se um polinômio linear (ordem 1) ou de maior ordem é necessário para representar seus dados.

Formato para os Casos de Teste

Formato da entrada:
<número k de coeficientes do polinômio>
<ponto 1 em X>
<ponto 1 em Y>
<ponto 2 em X>
<ponto 2 em Y>
...
<ponto k em X>
<ponto k em Y>

Formato da saída:
<coeficiente 1>
<coeficiente 2>
...
<coeficiente k>

Exemplo de Caso de Teste

Entrada fornecida:
4
0
10
1
7
3
-11
4
-14

Saída esperada:
1.000000
-6.000000

```
2.000000
10.000000
```

Informações Importantes

1) Um dos objetivos da disciplina de ICC2 é o aprendizado individual dos conceitos de programação. A principal evidência desse aprendizado está nos trabalhos, que são individuais neste curso. Você deverá desenvolver seu trabalho sem copiar trechos de código de outros alunos ou da Internet, nem codificar em conjunto. Portanto, compartilhem ideias, soluções, modos de resolver o problema, mas não o código.

1.1) O plágio vai contra o código de ética da USP;

1.2) Quando autores e copiadores combinam, estão ludibriando o sistema de avaliação, por isso serão utilizadas ferramentas de análise plágio tais como o MOSS (http://moss.stanford.edu/);

1.3) O trabalho em grupo e a cooperação entre colegas é em geral benéfico e útil ao aprendizado. Para ajudar um colega você pode lhe explicar estratégias e ideias. Por exemplo, pode explicar que é preciso usar dois loops para processar os dados, ou que para poupar memória basta usar uma certa estrutura de dados, etc. O que você não deve fazer é mostrar o seu código. Mostrar/compartilhar o código pode prejudicar o aprendizado de seu colega:

1.3.1) Depois seu colega ver seu código, será muito mais difícil para ele imaginar uma solução original e própria;

1.3.2) O seu colega não entenderá realmente o problema: a compreensão passa pela prática da codificação e não pela imitação/cópia.

1.4) Um colega que tenha visto a sua solução pode eventualmente divulgá-la a outros colegas, deixando você numa situação muito complicada, por tabela;

1.5) O texto acima foi baseado e adaptado da página http://www.ime.usp.br/~mac2166/plagio/, da qual recomendo a leitura completa.

2) Todos os códigos fontes serão comparados por um (ou mais) sistema(s) de detecção de plágio, e os trabalhos com alta similaridade detectada terão suas notas zeradas, tanto aqueles relativos ao código de origem quanto do código copiado. A detecção de plágio será reportada à Seção de Graduação para providências administrativas;

3) A avaliação incluirá a porcentagem de acertos verificada pelo Sistema de Submissão de Trabalhos e também a análise do seu código, incluindo identação, comentários, bom uso da memória e práticas de programação. Portanto faça seu código com cuidado, da melhor forma possível.

Sobre o sistema de submissão:

1. Seu código deverá incluir arquivo fonte .c .h e Makefile – todos os arquivos deverão obrigatoriamente conter no início um comentário com seu nome, número USP, turma e data da entrega do trabalho;

2. A data/hora de entrega do trabalho é aquela estipulada no sistema. Trabalhos entregues por email NÃO serão aceitos, mesmo que dentro da data/hora estipulada. Faça seu trabalho com antecedência para evitar entregar em cima da hora e ter problemas de submissão;

2.1. A submissão é de responsabilidade do aluno, e os problemas comuns à entrega próxima ao fechamento do sistema também. Portanto: problemas de acesso à rede não serão aceitos como

desculpa para entrega por email ou fora do prazo;

3. A compilação e execução do código é feita no sistema pelos comandos:

make all
make run

4. A saída do seu programa deve ser exatamente igual à saída esperada, incluindo: espaços em branco, quebras de linha e precisão decimal;

5. Há um limite em segundos para a execução dos casos de teste e um limite de memória total para ser utilizado. Você deverá gerenciar bem o tempo de execução e o espaço ocupado para que seu programa fique dentro desses limites, para evitar uso excessivo, pois o sistema irá invalidar o caso em que o limite foi excedido;

6. Ao enviar, aguarde sem recarregar a página nem pressionar ESC, para obter a resposta. Caso demore mais do que 2 minutos para dar uma resposta, feche e abra novamente a página do servidor. Verifique se seu programa tem algum problema de entrada de dados (ou se tem algum loop infinito ou parada para pressionamento de tecla). Caso não tenha, aguarde alguns minutos e tente novamente;

7. O erro de “Violação de Memória” significa acesso indevido a arranjo ou arquivo. Use compilação com g e o programa valgrind para tentar detectar a linha de código com erro.

7.1. Exemplo 1 de violação de memória:

```
int **mat = (int **)malloc(sizeof(int *) * 3);
mat[0] = (int *)malloc(sizeof(int)*10);
for (i = 1; i < 3; i++) {
   free(mat[i]);
}
// apenas a posicao 0 de mat foi alocada as outras nao
// portanto esta liberando regiao nao alocada
// gerando violacao de memoria
```

7.2) Exemplo 2 de violação de memória:

```
int B[5] = {5, 6, 7, 8, 9};
int N = 5;
int *A = malloc(N*sizeof(int));
int j = 0, i = 1; // 'j' inicializado em 0, 'i' inicializado em 1
while (j < N){
A[i] = B[j]; // 'j' e 'i' sao indices diferentes
j++;         // quando j = (N  1) i = (N  1)+1 e
i++;
     // havera escrita indevida em A
}
```