EP 2

Problema 1280 – Curvy Little Bottles

Grupo 16

Arthur Paulucci Carnieto	8921047
Higor Lopes da Silva	8921760
Marcello Malagoli Ziravello	8921242
Matheus Mendes de Sant'Ana	8921666

Desafios de Programação I
ACH2107 – Turma 03

SÃO PAULO
2017

1) Variáveis

1.1) Principais estruturas globais

coeficientes (tipo LinkedList<Double>) – Estrutura que será preenchida, em cada caso de teste, com os coeficientes do polinômio. Dada uma posição **p** da lista, o nó correspondente a essa posição possuirá o valor do coeficiente de grau **p**. Exemplo: dado o polinômio $\mathbf{x^2 + 3}$, o nó da **posição 0** (termo independente de x) conterá o valor **3**, o nó da **posição 1** (coeficiente que multiplica x^1) conterá o valor **0** e o nó da **posição 2** (coeficiente que multiplica x^2) conterá o valor **1**.

– **Exemplo de busca**: Através da chamada coeficientes.get(2) obteríamos o coeficiente de grau 2 (coeficiente que multiplica x^2 no polinômio). Voltando ao mesmo exemplo anterior do polinômio $\mathbf{x^2 + 3}$, ao realizarmos a chamada coeficientes.get(2) receberíamos o valor **1**.

– **Exemplo de inserção**: Para criarmos o polinômio $\mathbf{x^2 + 3}$, seria necessário realizar as chamadas coeficientes.add(3), coeficientes.add(0) e coeficientes.add(1) **nesta ordem**. Percebe-se que os nós são inseridos na posição da lista de acordo com a ordem de chamada, começando a partir da posição 0.

coeficientesAoQuadrado (tipo Map<Integer, Double>) – Será preenchida, através do método **preencherPolinomioAoQuadrado**(), com o valor dos coeficientes do polinômio elevado ao quadrado (para cada caso de teste). A estrutura **coeficientesAoQuadrado** depende do pressuposto que a estrutura **coeficientes** seja inicializada antes. Tendo como exemplo o polinômio $\mathbf{x + 3}$, seu respectivo polinômio ao quadrado seria $(x+3)^2$ resultando no polinômio $\mathbf{x^2 + 6x + 9}$. Continuando no mesmo exemplo (polinômio ao quadrado $x^2 + 6x + 9$), a estrutura teria a seguinte configuração: [f(0) → 9], [f(1) → 6] e [f(2) → 1], onde f é uma função *hash* qualquer: as **chaves** (tipo Integer) da estrutura *hash* são o **grau** de um termo do polinômio, as quais levam ao **valor** desse termo do polinômio (tipo Double).

– **Exemplo de busca**: Através da chamada coeficientesAoQuadrado.get(1) obteríamos o coeficiente de grau 1 (coeficiente que multiplica x^1 no polinômio ao quadrado). Voltando ao mesmo exemplo anterior do polinômio $x + 3$ cujo polinômio ao quadrado seria $\mathbf{x^2 + 6x + 9}$, ao realizarmos a chamada coeficientesAoQuadrado.get(1) receberíamos o valor **6**.

– **Exemplo de inserção**: Para inserirmos o polinômio $\mathbf{x^2 + 6x + 9}$ (ou seja, o polinômio $x + 3$ elevado ao quadrado) na estrutura, seria necessário executarmos as chamadas coeficientesAoQuadrado.put(0,9), coeficientesAoQuadrado.put(1,6) e coeficientesAoQuadrado.put(2,1) – não necessariamente nessa ordem.

marcas (tipo LinkedList<Double>) – Durante a execução do algoritmo, esta estrutura será preenchida com as distâncias do eixo “**x**” que dividem o volume da garrafa em partições de volume “**v**” (“**v**” é dado na entrada de cada caso de teste). Após a execução do algoritmo, se essa estrutura estiver vazia será exibida a mensagem “insufficient volume”, do contrário serão exibidos todos os elementos da lista.

1.2) Principais variáveis auxiliares

xLow (tipo double) – Valor no eixo X da base da garrafa.

xHigh (tipo double) – Valor no eixo X do topo da garrafa.

distanciaX (tipo double) – Para uma determinada partição da garrafa, essa variável armazena a distância entre o valor do **eixo x** dessa partição menos o valor **xLow**.

volumeTotal (tipo double) – Volume total da garrafa.

volumeParticoes (tipo double) – Volume que cada partição da garrafa deve conter.

volumePartidoAcumulado (tipo double) – Volume acumulado de todas as partições que foram criadas durante a execução do algoritmo. Exemplo: para um determinado caso de teste que tenha o volume das partições igual a 25 dado na entrada, o volume partido acumulado para a primeira partição será 25, para a segunda partição será 50 (duas vezes 25), para a terceira partição será 75 (três vezes 25) e assim sucessivamente.

2) Algoritmo

2.1) Inicialização das variáveis

Para cada caso de teste, as estruturas "**coeficientes**", "**coeficientesAoQuadrado**" e "**marcas**" são reinicializadas. A variável global "**grauPolinomio**" também é reinicializada para cada caso de teste.

Assim como explicado na seção 1.1, a estrutura "coeficientes" será preenchida com os coeficientes do polinômio em suas respectivas posições, enquanto a estrutura "coeficientesAoQuadrado" será preenchida com os coeficientes do mesmo polinômio, porém este elevado ao quadrado. A variável "grauPolinomio" receberá o valor do grau do polinômio. Segue abaixo um exemplo de execução do algoritmo:

Entrada:

1

4 1

0.0 12.0 10

Como ficam as principais variáveis:

grauPolinomio: 1

coeficientes: 4, 1

coeficientesAoQuadrado: [f(0)→16], [f(1)→8], [f(2)→1]

xLow: 0

xHigh: 12

volumeParticoes: 10

Pela 1ª linha da entrada percebemos que o grau do polinômio será **1**; a 2ª linha nos informa que, através dos coeficientes, o polinômio será **4 + x** (então o polinômio ao quadrado será $\mathbf{x^2 + 8x + 16}$, representado na estrutura "coeficientesAoQuadrado"); a 3ª linha nos informa que **xLow** será 0, **xHigh** será 12 e o **volume das partições** (variável "**volumeParticoes**") será 10. A função "f" na estrutura "coeficientesAoQuadrado" é uma função *hash* qualquer.

2.2) Encontrando a solução

Definiremos uma função correspondente ao polinômio, igualando y ao valor do polinômio, exemplo: se o polinômio é **x + 4**, a **função** correspondente será **y = x + 4**. A parte principal do algoritmo é baseada no cálculo do volume de um sólido de revolução: o volume de um sólido de revolução é igual a $\boldsymbol{\pi A^2}$, onde "**A**" é a área que será rotacionada. Para calcularmos o valor de "A", utilizaremos a integral da função correspondente entre **xLow** e **xHigh**. A fórmula do volume do sólido será então igual a $\boldsymbol{\pi * \int [f(x)]^2}$, ou seja, π vezes a integral entre xLow e xHigh da função ao quadrado. Para o cálculo da integral, utilizamos o método de aproximação de Simpson.

No arquivo do EP existe o método "volume", recebendo os parâmetros "**esq**", "**dir**" e "**n**"; nos retornando a aproximação do volume do sólido para cada caso de entrada. Os parâmetros "**esq**" e "**dir**" são os valores do intervalo da integral, enquanto "**n**" é o número escolhido de partições para o método de Simpson – usamos como padrão n = 1000.

Grosso modo, a saída consistirá em exibir **2 conjuntos de valores**: o **volume total do sólido** e as **distâncias entre o valor x de cada partição do sólido e xLow** (xLow será sempre fixo para cada caso de teste). Para encontrarmos o volume total do sólido será simples: invocar o método "**volume**" com os seguintes parâmetros **esq** = xLow, **dir** = xHigh e **n** = 1000.

Para encontrarmos os valores de **x** de cada partição do sólido, será necessário procurar o **x** tal que o intervalo entre **x** e **xLow** (este último dado na entrada) possua o volume acumulado da partição atual, exemplo: dado o volume de partição igual a 20 (recebido na entrada), a **partição 1** terá volume acumulado igual a 20, o volume acumulado da **partição 2** será 40 e da **partição 3** será 60. É possível encontrar esse **x** – com erro de aproximação pequeno – através de uma busca binária entre **xLow** e **xHigh**, adicionando então na estrutura "**marcas**" – para cada partição do sólido – a diferença entre o **x** encontrado e **xLow** (*).

(*) <u>**OBS:**</u> O algoritmo não permite que a estrutura "marcas" tenha mais do que 8 nós, parando a execução ao encontrar 8 partições. Se a busca binária não encontrar um valor aproximado para "x" – retornando um valor inválido – significa que não há volume suficiente no sólido para comportar o volume acumulado buscado.

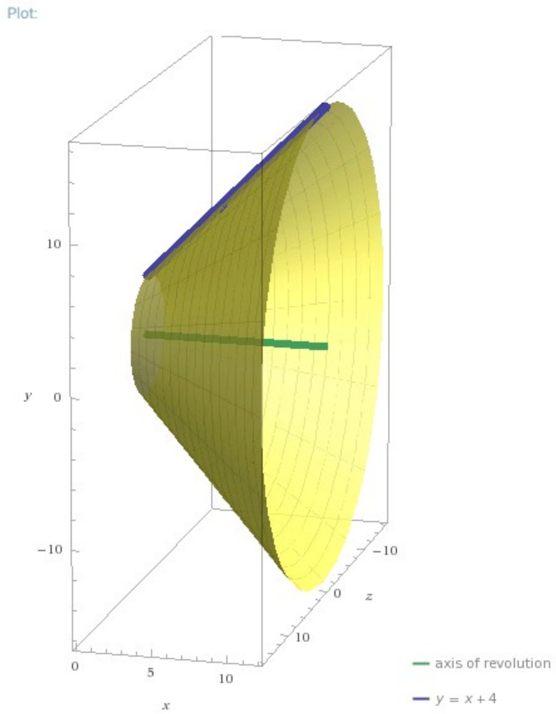

Fig.1 Sólido de revolução, entre **x=0** e **x=12**, para o polinômio **x + 4** (função **y = x + 4**)

3) Dificuldades encontradas

1ª) Encontrar o intervalo do eixo x que contivesse um determinado volume procurado (utilizamos esse algoritmo para encontrar os valores de x de cada partição da garrafa): o intervalo deveria iniciar em **xLow** e terminar em um outro **valor "x" procurado**. Inicialmente optamos por fazer uma **busca sequencial**, iniciando a partir de xLow e incrementando 0.000001 até encontrarmos o valor de x procurado. Após recebermos o resultado "Time limit exceeded", modificamos o algoritmo para realizar uma **busca binária** entre xLow e xHigh, utilizando incrementos ou decrementos de 0.000001 ao invés de realizarmos uma busca sequencial iniciando em xLow.

2ª) Assim como citado no tópico anterior, o incremento ou decremento de 0.000001 na busca binária foi proveniente de algum número pequeno com as seguintes propriedades: deveria fornecer precisão suficiente para não retornar "Wrong answer"; mas também deveria ser grande o suficiente para a busca não demorar muito a ponto de retornar "Time limit exceeded".

3ª) Número de divisões para aproximar a integral: deveria ser grande o suficiente para aproximar a integral precisamente, não retornando o resultado "Wrong answer"; mas não poderia ser tão grande de modo que gastasse muito tempo, retornando "Time limit exceeded". O valor escolhido foi de 1000 divisões.

4ª) Encontrar um método de aproximação para calcularmos a integral. Após recebermos os resultados "Time limit exceeded" e "Wrong answer" utilizando diferentes parâmetros no **método dos trapézios**, optamos por utilizar o método de **Simpson** que nos retornou o resultado "Accepted".