EP Opcional

Problema 1115 – Water Shortage

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) Principal estrutura global

Cisterna (classe interna) – Classe global interna, utilizada para cada instância de uma cisterna nos casos de teste. Possui 4 atributos semanticamente autoexplicativos:

1º) "**alturaBase**" contém a altura da base da cisterna.
2º) "**alturaTopo**" contém a altura do topo da cisterna.
3º) "**largura**" contém a largura da cisterna.
4º) "**profundidade**" contém a profundidade da cisterna.

1.2) Principais variáveis locais

cisternas (tipo LinkedList<Cisterna>) – Principal estrutura local. Armazena o conjunto de cisternas com seus respectivos valores de atributos.

volumeInjetado (tipo double) – Volume que será injetado no sistema de cisternas para cada caso de teste. Este valor é dado na entrada.

alturaBaseMaisBaixa (tipo double) – Altura da base mais baixa dentre todas as cisternas.

alturaTopoMaisAlto (tipo double) – Altura do topo mais alto dentre todas as cisternas.

patamares (tipo double[]) – Contém a **altura da base** e a **altura do topo** de todas as cisternas. Por exemplo, dado um caso de teste com as seguintes cisternas:
– **1ª cisterna** possui **altura da base = 2** e **altura do topo = 3**;
– **2ª cisterna** possui **altura da base = 1** e **altura do topo = 3**.
O vetor "patamares" terá a seguinte configuração: {2,3,1,3}.

patamaresSemRep (tipo double[]) – Possuirá o mesmo conteúdo do vetor "patamares", porém os valores estarão ordenados e sem repetições. Esse vetor será inicializado com todos os valores iguais a -1. Posteriormente, manterá nas posições à esquerda todos os valores ordenados e sem repetições do vetor "patamares", enquanto que nas posições à direita que sobrarem – devido à exclusão de valores repetidos – haverá apenas posições preenchidas com o valor **-1**. Voltando ao exemplo citado no tópico anterior, dado que a variável "patamares" teria a configuração {2,3,1,3}, o vetor "patamaresSemRep" apresentaria a seguinte

configuração: {1,2,3,-1}.

areaAcumuladaAtual (tipo double) – A cada **par de patamares consecutivos** do vetor "**patamaresSemRep**", teremos uma área acumulada resultante da seguinte soma: área da base (igual ao produto largura * profundidade) de todas as cisternas que satisfaçam a condição [1] **altura da base** <= **patamar atual inferior** e **altura do topo** >= **patamar atual superior**. Voltando ao exemplo anterior (dado nos tópicos "patamares" e "patamaresSemRep") no qual o vetor "patamaresSemRep" apresenta a configuração {1,2,3,-1}, teremos 2 pares de patamares consecutivos (pares com valor -1 são desprezados): **{1,2}** e **{2,3}**. Ainda retomando o exemplo:
– **1ª cisterna** possui **altura da base** = **2** e **altura do topo** = **3**;
– **2ª cisterna** possui **altura da base** = **1** e **altura do topo** = **3**.
Para o par {1,2} (isto é, patamar inferior = 1 e patamar superior = 2) apenas a 2ª cisterna satisfaz a condição [1], logo o valor da variável "areaAcumuladaAtual" para esse par será igual à área da base da 2ª cisterna. Para o par {2,3}, ambas as cisternas satisfazem a condição [1], logo o valor da variável "areaAcumuladaAtual" para esse par será igual à área da base da 1ª cisterna somada à área da base da 2ª cisterna.

volumeAcumuladoAtual (tipo double) – A cada par de patamares consecutivos do vetor "**patamaresSemRep**", será calculado o volume presente entre esses patamares através da multiplicação da variável "areaAcumuladaAtual" (tópico anterior) com a diferença entre o patamar atual superior e o patamar atual inferior. Explicando através de uma fórmula:
volumeAcumuladoAtual = (patamar atual superior – patamar atual inferior) * areaAcumuladaAtual.

alturaAcumulada (tipo double) – Será a variável imprimida nos casos gerais, indicando a altura que a água injetada no sistema alcançou. Caso a água ultrapasse a altura do topo mais alto dentre todas as cisternas (isto é, ultrapasse o valor da variável "alturaTopoMaisAlto"), será exibida a mensagem "OVERFLOW".

2) Algoritmo

Para cada caso de teste, as estruturas "**cisternas**", "**patamares**" e "**patamaresSemRep**" são reinicializadas. As variáveis "**volumeInjetado**", "**alturaBaseMaisBaixa**" e "**alturaTopoMaisAlto**" também são reinicializadas para cada caso de teste.

Vale lembrar que a altura do topo de uma cisterna não é dada diretamente na entrada, ela será calculada então pela seguinte soma: **altura da base da cisterna** + **altura da cisterna**. Segue abaixo um exemplo de execução do algoritmo:

Entrada:

1

4

3 4 5 5

0 7 5 5

4 2 5 5

0 9 5 5

400

Como ficam as principais variáveis:

patamares: {3,7,0,7,4,6,0,9}

patamaresSemRep: {0,3,4,6,7,9,-1,-1}

alturaBaseMaisBaixa: 0

alturaTopoMaisAlto: 9

volumeInjetado: 400

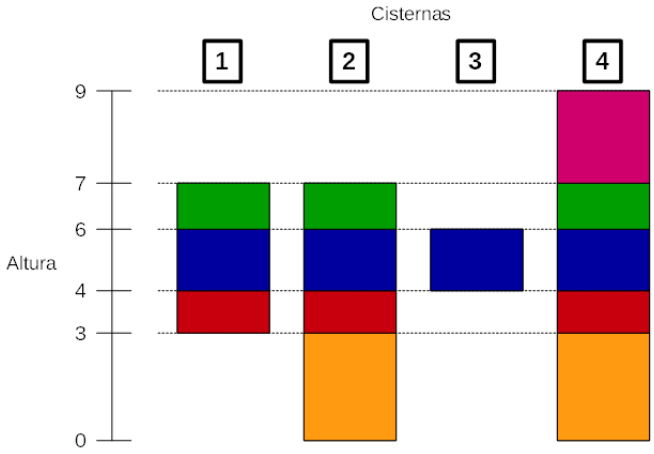

Fig.1 – Preenchimento dos patamares para a entrada dada: seções das cisternas com a mesma cor estão no mesmo par de patamares.

Simplificamos a entrada para maior compreensão, criando 4 cisternas com mesma largura (igual a 5) e mesma profundidade (igual a 5), logo a área da base de todas as cisternas será igual a 25 (ou seja, largura * profundidade). A 1ª linha da entrada nos remete a apenas 1 caso de teste, a 3ª linha nos informa que existirão 4 cisternas para o caso de teste; as 4 linhas posteriores à 3ª nos fornecem as dimensões das cisternas e a última linha é o valor do volume injetado no sistema.

Se a variável "**volumeInjetado**" for **menor ou igual a zero**, o resultado da altura alcançada será igual a **zero**. Do contrário, caso o número de cisternas seja menor ou igual a zero ou só exista um patamar na variável "**patamaresSemRep**", será imprimida a mensagem "OVERFLOW" pois não existem cisternas que

possam armazenar qualquer volume. Para os casos gerais, serão preenchidos os volumes das cisternas entre os pares de patamares consecutivos do vetor "**patamaresSemRep**", começando da posição 0 até a última posição desse vetor, exemplo: na Fig.1 procurar-se-á preencher os volumes amarelo, vermelho, azul, verde e rosa respectivamente.

Para cada um desses pares preenchidos (completamente ou parcialmente), será acrescentada a altura de acordo com o volume de água que foi adicionado. Durante o preenchimento de um determinado par de patamares (por exemplo o volume amarelo da Fig.1), ocorrerão 2 condições:

1ª) Se o par de patamares atual é o **último** (seção rosa da Fig.1). Se o volume desse par for menor do que o volume que ainda resta para ser adicionado, então será exibida a mensagem "OVERFLOW" e o algoritmo termina. Do contrário será exibida a altura final, equivalente à soma da **altura acumulada dos pares de patamares anteriores** + **[volume que ainda resta / área acumulada atual]** (entre colchetes está representada a altura restante deste último par) e o algoritmo termina.

2ª) Se o par de patamares atual ainda **não é o último** (exemplo: qualquer seção da Fig.1 exceto a rosa). Se o volume neste par for menor que o volume que resta para ser adicionado, o volume que resta receberá um novo valor igual ao seu próprio valor menos o volume do par atual; a altura acumulada será somada com a altura do par atual (exemplo: a altura do par vermelho na Fig.1 é igual a 1) e o algoritmo continua. Do contrário será exibida a altura final, equivalente à soma da **altura acumulada dos pares de patamares anteriores** + **[volume que ainda resta / área acumulada atual]** (entre colchetes está representada a altura restante deste último par) e o algoritmo termina.

3) Dificuldades encontradas

1ª) Criar uma estrutura para armazenar os atributos de cada cisterna. A primeira ideia pensada foi criar uma lista ligada de objetos do tipo "Double" para cada um dos 4 atributos de uma cisterna, ou seja, criar uma lista ligada para cada um dos seguintes atributos: altura da base, altura do topo, largura e profundidade. Para tornarmos o código mais simples e mais compreensível, optamos por criar a classe interna "**Cisterna**" que seria responsável por cada "cisterna" do mundo real, armazenando os seus 4 atributos. Após definido o tipo de objeto "Cisterna", criamos a lista ligada "**cisternas**" que conterá as instâncias desse tipo de objeto para cada caso de teste.

2ª) Durante a execução das primeiras versões do EP, utilizávamos o método **nextDouble()** do *Scanner* para recuperarmos os valores da entrada correspondentes aos números de ponto flutuante. Utilizando esse método, recebíamos exceções do tipo "InputMismatchException" em testes locais, por causa das casas decimais dos números estarem separadas pelo caractere "." e não pelo caractere ",". Dito isso, trocamos as chamadas "**sc.nextDouble()**" por "**Double.parseDouble(sc.next())**".