Relatório – Os três jarros do sheik

Arthur Pereira Viegas

Escola Politécnica – PUCRS

Resumo: Este artigo descreve uma alternativa para solucionar o problema proposto na disciplina de Algoritmos e Estrutura de Dados II, que consiste em descobrir quantos movimentos são necessários para mover a água entre os jarros e obter um resultado específico. A modelagem do problema e o processo da solução é apresentado, juntamente com o pseudocódigo dos algoritmos mais importantes. Além disso, proponho um caminho para dar continuidade ao estudo do problema. Por fim, serão apresentados os resultados obtidos nos seis casos de teste disponibilizados pelo professor.

Enunciado do problema

No contexto do problema, estou prestando assessoria para um sheik de um emirado distante. O sheik deseja que eu resolva um problema que lhe foi passado por um gênio a muitas gerações. Dados 3 jarros de água, devemos transferir a água dentro dos jarros entre eles respeitando as seguintes regras:

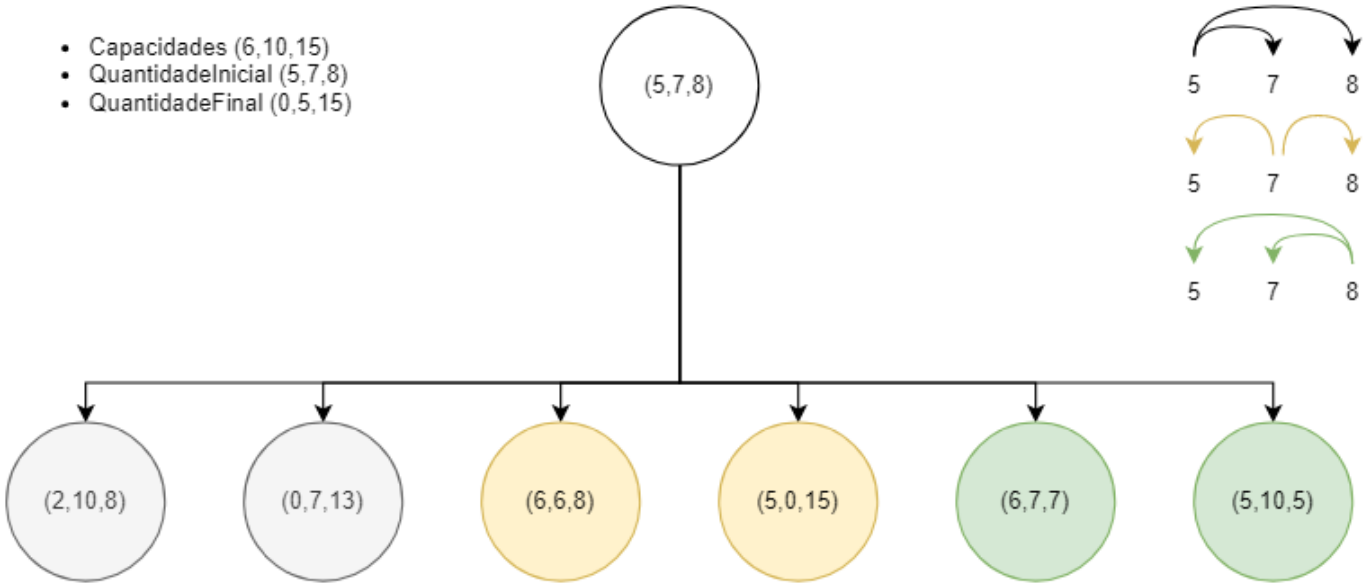
1. É proibido jogar água fora.
2. É proibido pegar água de uma fonte.
3. Só podemos ou esvaziar um jarro completamente, ou encher outro até a borda.

Além disso, nós recebemos mais algumas informações sobre os jarros, seriam estas:

- As capacidades de água de cada jarro (com limite máximo de 40 litros).
- As quantidades inicias de água de cada jarro.
- As quantidades desejadas em cada jarro para completar o desafio.

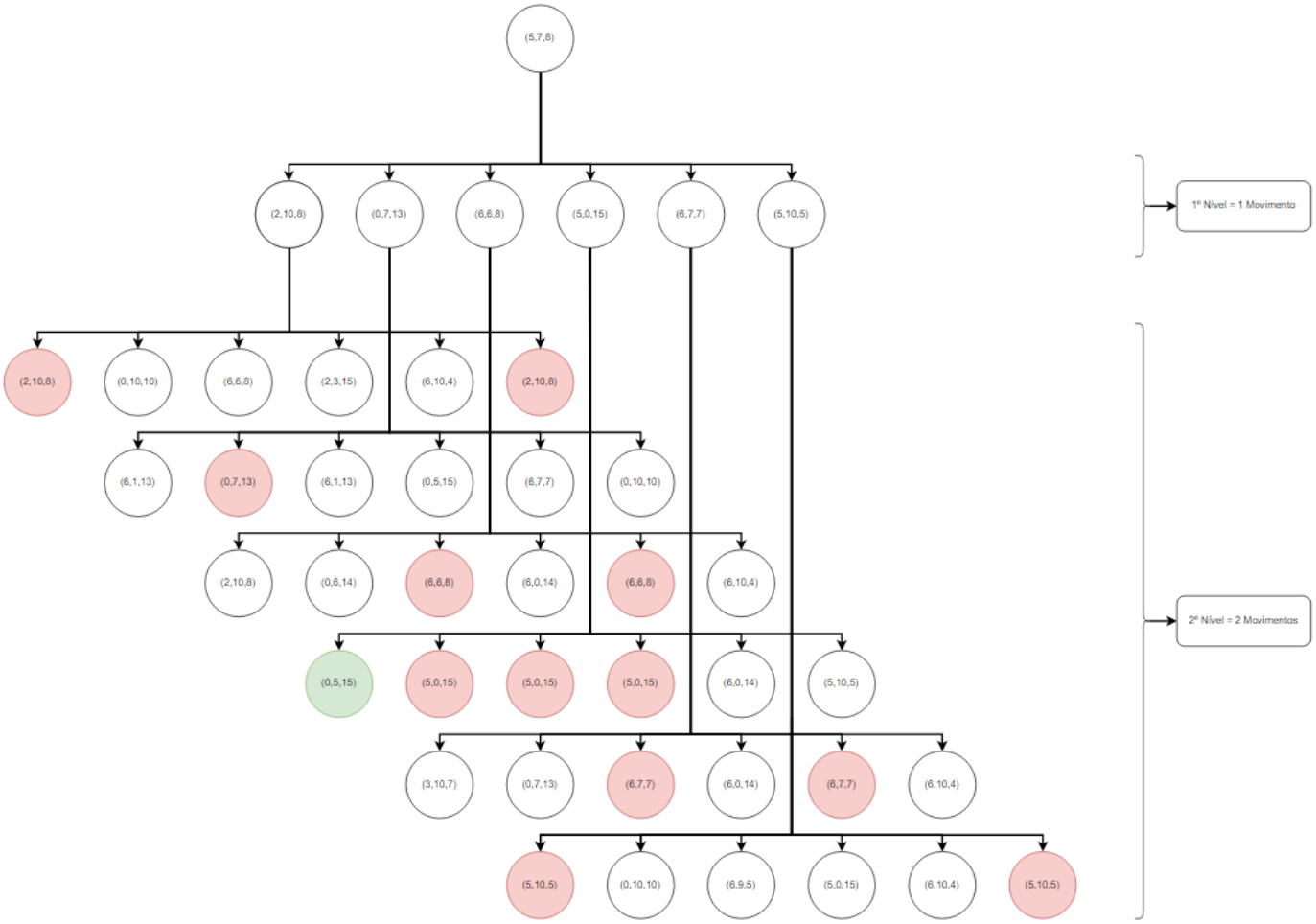
Etapas da solução, apresentando exemplos e algoritmos

Desde o primeiro momento do desenvolvimento do processo, tentei resolver através de uma Árvore Não-Binária. A ideia principal é que a partir de um nodo original, sejam criados 6 novos nodos a partir da movimentação de toda a água possível de um JarroOrigem para um JarroDestino, conforme a ilustração abaixo:



Nesta próxima etapa cada nodo é adicionado em uma lista de nodos e um método checa se algum dos nodos atuais é o Nodo que contém os jarros com a quantidade final desejada.

Seguindo este exemplo, o nodo final ainda não foi encontrado, então, para cada nodo criado no nível anterior da árvore nós criamos mais 6 nodos, através de recursão.



Aqui neste exemplo, há diversos nodos que não serão adicionados na lista de novos nodos, pois eles são nodos inválidos, sendo alguns nodos ou repetidos ou não sendo possíveis de realizar o despejo da água.

Como atingimos o nodo desejado em 2 movimentos, então após o programa comparar o nodo criado com o final, o programa se encerra.

Aqui abaixo alguns algoritmos do programa:

Main do programa:

```java
public static void main(String[] args) throws FileNotFoundException, UnsupportedEncodingException {
    //instanciacao do scanner
    Scanner scan = new Scanner(System.in);
    Scanner in = new Scanner(System.in);

    System.out.printf("Digite o nome do arquivo: ");

    String path = scan.nextLine();//Caminho do arquivo

    try(BufferedReader br = new BufferedReader(new FileReader(path))){
        String line = br.readLine();// Leitor de linha

        //Instanciação dos objetos
        while(line != null){
            String[] vetor1 = line.split(" ");
            Integer Capacidade1 = Integer.parseInt(vetor1[0]);
            Integer Capacidade2 = Integer.parseInt(vetor1[1]);
            Integer Capacidade3 = Integer.parseInt(vetor1[2]);
            line = br.readLine();//Le uma linha

            String[] vetor2 = line.split(" ");
            Integer AguaAtual1 = Integer.parseInt(vetor2[0]);
            Integer AguaAtual2 = Integer.parseInt(vetor2[1]);
            Integer AguaAtual3 = Integer.parseInt(vetor2[2]);
            line = br.readLine();

            String[] vetor3 = line.split(" ");
            Integer QuantidadeFinal1 = Integer.parseInt(vetor3[0]);
            Integer QuantidadeFinal2 = Integer.parseInt(vetor3[1]);
            Integer QuantidadeFinal3 = Integer.parseInt(vetor3[2]);

            Node NodoFinal = new Node(QuantidadeFinal1,QuantidadeFinal2,QuantidadeFinal3);

            Jarro Jarro1 = new Jarro(Capacidade1, AguaAtual1, QuantidadeFinal1, 1);
            Jarro Jarro2 = new Jarro(Capacidade2, AguaAtual2, QuantidadeFinal2, 2);
            Jarro Jarro3 = new Jarro(Capacidade3, AguaAtual3, QuantidadeFinal3, 3);

            ListaJarros.add(Jarro1);
            ListaJarros.add(Jarro2);
            ListaJarros.add(Jarro3);

            PrintWriter writer = new PrintWriter("Saida.txt", "UTF-8");
            writer.println(NodoFinal);
            writer.close();
        }


    }catch(IOException e){
        System.out.println("Erro: " + e.getMessage());
    }

}//Fim da Main
```

Atualmente a função main do programa, lê um arquivo de texto com 3 linhas (através do BufferedReader), quebra as linhas em vetores de Strings e instancia os jarros com as informações dos vetores de Strings. No final do programa, ele escreve o Nodo Final desejado em um arquivo de texto

Algoritmo de despejo de água:

O despejo de água funciona com a união do método do jarro de origem enviando a água contida nele atualmente, para um método receptor, no jarro de destino, que faz os cálculos das capacidades e devolve a quantidade excedente (se houver), para o jarro de origem.

```java
public void despejaAgua(Jarro jarrox) {
        AguaAtual = jarrox.setAguaAtual(AguaAtual);
    }

public Integer setAguaAtual(Integer Quantidade) {
        if(Quantidade > 0){
            if((Quantidade+AguaAtual)>Capacidade){//Se tiver Overflow
                Sobra = (AguaAtual+Quantidade)-Capacidade;
                AguaAtual = Capacidade;
                return Sobra;
            }
            AguaAtual=AguaAtual+Quantidade;//Se não ultrapassar Capacidade
            return Sobra;
        }
        return Sobra;
    }
```

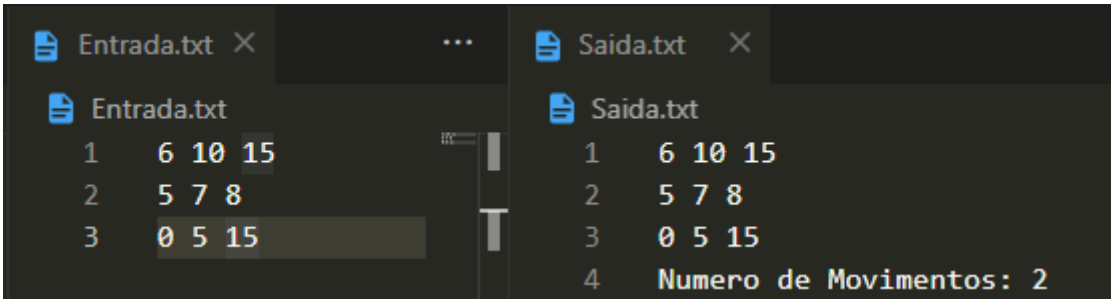
Resultados dos casos de teste

No momento atual não consegui finalizar o algoritmo que faz tudo automaticamente, sendo possível atingir o nodo final apenas transferindo a água manualmente, contudo já possuo o algoritmo para a execução automática

Sendo ele:

1. Cria Nodo Pai
2. Gera 6 Nodos.
   - Nodo1 = Jarro1.despejaAgua(Jarro2)
   - Nodo2 = Jarro1.despejaAgua(Jarro3)
   - Nodo3 = Jarro2.despejaAgua(Jarro1)
   - Nodo4 = Jarro2.despejaAgua(Jarro3)
   - Nodo5 = Jarro3.despejaAgua(Jarro1)
   - Nodo6 = Jarro3.despejaAgua(Jarro2)
3. Adiciona Nodos em Lista de Novos Nodos
4. Checa Se um dos Nodos Novos é a solução (Se não: Contador de Movimentos++)
5. Volta para a segunda etapa tendo o Nodo1 como “Nodo Pai”

Entrada e Saída com o caso 1 manualmente:



Conclusões

Gostei de desenvolver o programa, a ideia dele é muito boa, contudo, foi infeliz de minha parte não conseguir terminar o construtor da árvore, mas fora isto o trabalho foi bom para o meu desenvolvimento profissional e pessoal. Mesmo após a entrega do trabalho, continuarei o desenvolvimento do programa, para fins de estudo, e deixo aqui o link para o repositório.