Exercício 3

Exploração em Marte

Unidade Curricular de
Laboratórios de Programação

2020/2021

Exercício: *Exploração em Marte*

Objetivos

- Resolução de problemas utilizando programação

Descrição do problema

Um novo *rover* chamado *Esperança* aterrou no planeta Marte e a sua missão é atualizar um dos módulos de software que permite fazer medições e análise a rochas.

Durante a aterragem o braço robótico que manuseia as rochas sofreu uma avaria, e agora, quando apanha rochas, estilhaça-as em pequenos retângulos. Por sorte,

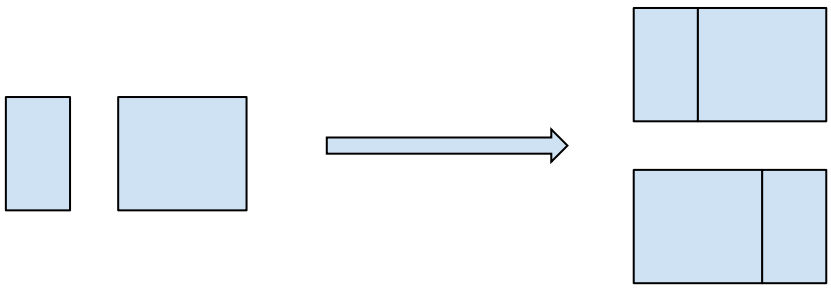
- as rochas desta localização de Marte têm todas uma **forma retangular de lados inteiros,** e área inteira;
- é possível ver pela câmara frontal a **largura original** da rocha;
- sabe-se a largura e altura de cada um dos estilhaços de rocha retangular;
- sabe-se que os estilhaços permanecem na sua orientação original, pelo que não precisam de ser rodados e têm lados inteiros.

PRIMEIRO OBJETIVO
O seu primeiro objetivo é **calcular a altura original da rocha antes de ter partido**.

Exemplo: Suponha que recebeu a informação que

- a largura original da rocha é 3
- a rocha partiu em 2 pedaços, com dimensões 2x2 e 1x2 (*largura x altura)*

Neste caso, a única altura que a rocha original poderia ter é 2, apesar de existirem duas configurações possíveis.



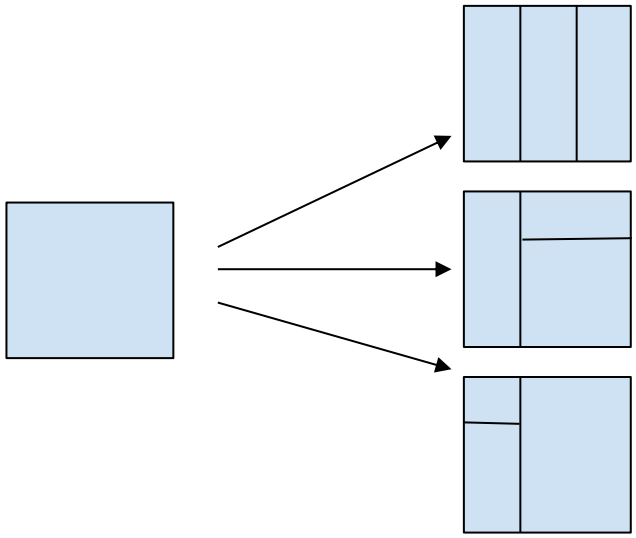
Observação: para calcular a altura original não precisa de determinar a posição original de cada pedaço -- isso seria um problema bastante difícil de resolver!

Não queremos certamente que o nosso *rover* tenha algum problema durante a execução do novo módulo de análise a rochas, portanto temos de garantir que irá funcionar corretamente. Para isto, vamos gerar testes de rochas para as quais sabemos a altura.

SEGUNDO OBJETIVO
O seu segundo objetivo é, dados um número de pedaços **N** e uma rocha original com dada altura e largura, separá-la em **N** pequenos retângulos de lado inteiro de modo a poder testar o seu procedimento para análise de rochas. Esta geração deve ser aleatória.

Exemplo: Suponha que quer criar um teste onde
- as dimensões originais da rocha são 3x3;
- quer quebrar a rocha em 3 pedaços.

A sua solução deve gerar aleatoriamente uma configuração possível. São mostradas, por exemplo, 3 configurações possíveis:



A informação dos pedaços de rocha é dada ao *rover* através de ficheiros `.rock`, que o procedimento de teste deverá gerar também.

Sugestão: Comece com o retângulo original e divida-o. Depois, escolha um rectângulo aleatoriamente e repita o processo até ter o número de retângulos necessário.

Formato *.rock*

Os ficheiros *.rock* que contêm a informação sobre os pedaço de rocha seguem o seguinte formato:

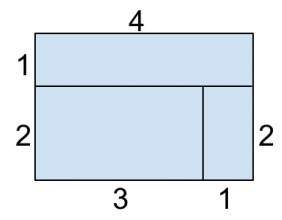
```
LARGURA_ORIGINAL
NUMERO_PEDAÇOS
Largura1 Altura1
...
...
LarguraNUMERO_PEDAÇOS AlturaNUMERO_PEDAÇOS
```

onde `LarguraX`, `AlturaX >= 1`

Por exemplo, um ficheiro *broken.rock* com

```
4
3
3 2
1 2
4 1
```

corresponde a uma rocha como a da figura ao lado.



Observe que poderia rearranjar os pedaços de outra maneira (sem os rodar), mas isso não faria diferença para obter a altura original da rocha, que neste caso é 3.

Antes de Começar

Deverá descarregar o ficheiro `alunosExercicio3.zip` e importar o ficheiro para o Eclipse utilizando o menu **File-> Import -> Existing Projects into Workspace -> From Archive**.

O que fazer

Deverá:

programar a classe **RockAnalysis** com

- um método estático **`int computeOriginalHeight(String filename)`** que calcula a altura original da rocha, dado o nome do ficheiro que se assume seguir o formato *.rock* com as informações dos pedaços de rocha

completar a classe **RockTestGenerator** com

- um método estático **`String generate(int numberOfPieces, int width, int height)`** que recebe um número de pedaços, e uma altura e largura de uma rocha, e devolve uma **`String`** com uma descrição de uma rocha separada em pedaços no formato *.rock.* Assume-se que **`1<= numberOfPieces <= min(5000000, width * height)`** e **`1 <= width, height <= 10000`**

Além da documentação usual, deve **descrever detalhadamente a metodologia que utilizou para implementar os métodos descritos acima** no comentário do método.

Pode implementar outras classes auxiliares. Alguns dos testes fornecidos têm limite de tempo.

Entrega

Deve entregar os ficheiros `RockAnalysis.java` e `RockTestGenerator.java`, tais como outras classes auxiliares que tenha implementado.