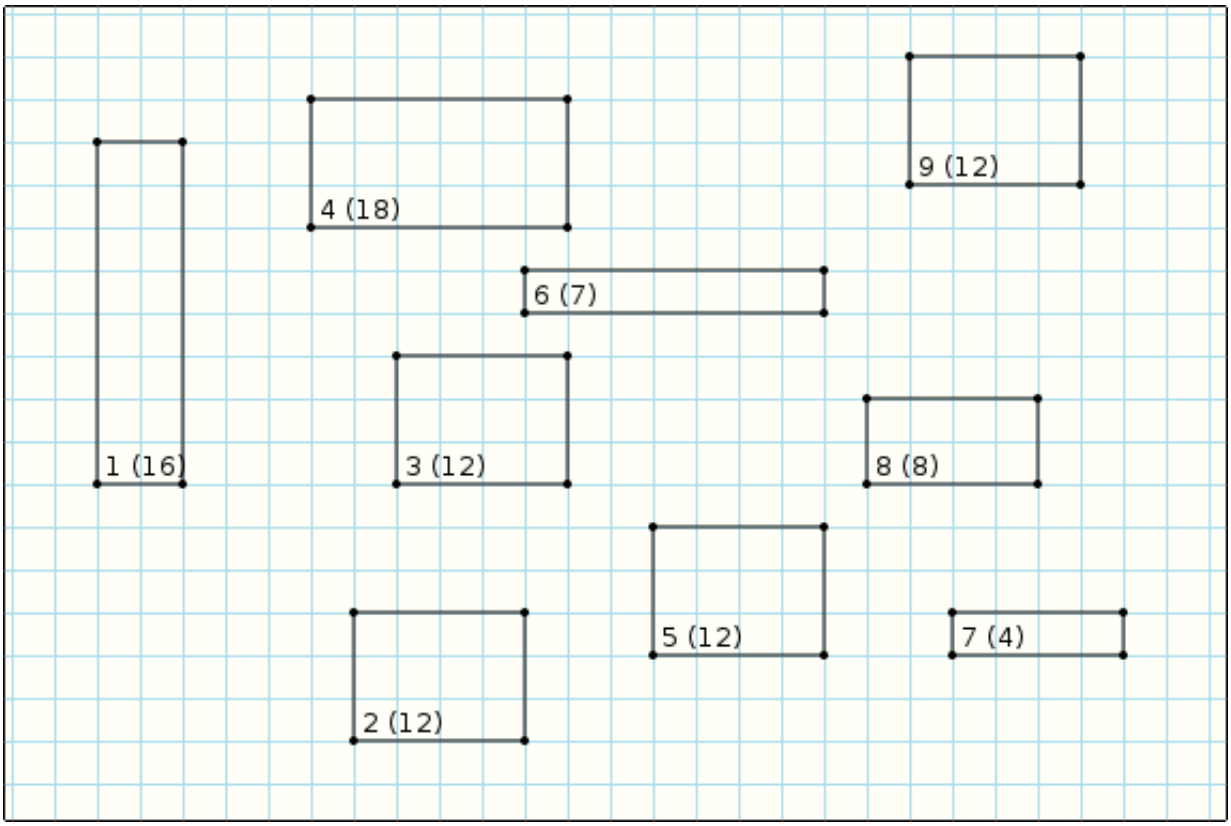
MC346 - Paradigmas de Programação: Programação Funcional: Desafio

Criada: 2016-04-24

Projeto Haskell

Imagine que tenhamos vários retângulos desenhados na tela, por exemplo:



Nesta figura, cada retângulo vem acompanhado de um rótulo localizado dentro do retângulo, abaixo à esquerda, e também um número entre parânteses que é a área do retângulo medida em quadradinhos azuis da rede subjacente.

Seu desafio envolvendo a linguagem Haskell será construir um subconjunto dos retângulos dados que sejam compatíveis e com a maior área total possível. O que são retângulos compatíveis? Dois retângulos são compatíveis quando um deles tem as coordenadas X todas menores que as coordenadas X do outro, e também as coordenadas Y menores que as coordededadas Y do outro, sem intersecção.

Vejamos alguns exemplos. Na figura acima, as coordenadas X são medidas no eixo horizontal, da esquerda para a direita. As coordenadas Y, por sua vez, são medidas verticalmente, de cima para baixo. Esta convenção é comum em gráficos de computador que são feitos para exibição em tela, embora seja contrária à convenção matemática, usado por exemplo em geometria analítica, onde as coordenadas Y crescem de baixo para cima. Mas vamos utilizar a convenção

de gráficos de computador neste documento.

Os retângulos 4 e 5 são compatíveis, pois 5 tem coordenadas X e Y maiores que as de 4. Porém, 4 e 9 não são compatíveis, pois existe intersecção nas coordenadas Y destes dois retângulos. Por outro lado, 2 e 8 também não são compatíveis, pois, apesar de não haver intersecção em suas coordenadas (nem X, nem Y), o retângulo 2 possui as maiores coordenadas Y, enquanto que o 9 possui as maiores coordenadas X entre os dois retângulos. Logo, nenhum dos dois tem as maiores coordenadas em ambos os eixos.

Você terá que juntar um subconjunto dos retângulos dados que tenha as seguintes propriedades:

- são compatíveis entre si, isto é, cada um é compatível com todos os demais do subconjunto;
- a área somada dos retâgulos de seu subconjunto é a maior possível.

No caso do exemplo dado na figura, a solução é o subconjunto {4,5}. Este subcunjuto é formado de retângulos compatíveis entre si, sua área total é 30 e não há outro subconjunto compatível com área total maior. Neste caso a resposta teve apenas dois retângulos, mas em geral as respostas poderão ter mais retângulos (ou até mesmo um único retângulo).

Tente fazer isto da forma mais eficiente possível, pois sua nota vai depender, em parte, do desempenho de seu programa (tempo de processamento). Para ser eficiente, seu programa precisará de estruturas de dados mais sofisticadas do que as listas e tuplas disponíveis nativamente na linguagem, e o desafio destina-se a avaliar quão bem você consegue implementar estruturas avançadas em estilo funcional.

Sobre a entrada

Você deverá receber os retângulos do dispositivo de entrada padrão, a ser lido como um arquivo de texto. Retângulos serão dados como um identificador inteiro seguido de 4 inteiros indicando as coordenadas dos vértices extremos do retângulo, na forma:

```
id xmin xmax ymin ymax
```

onde garantidamente teremos `xmin` < `xmax` e `ymin` < `ymax`.

Será dada como entrada uma lista de tais retângulos, um por linha, em uma ordem qualquer, sendo que garantidamente os identificadores não se repetem.

Dois retângulos A = `(xminA, xmaxA, yminA, ymaxA)` e B = `(xminB, xmaxB, yminB, ymaxB)` são compatíveis quando `xmaxA < xminB` e `ymaxA < yminB`, ou quando `xmaxB < xminA` e `ymaxB < yminA`.

No caso da figura acima, o arquivo de entrada poderia ter sido o seguinte:

```
1 3 5 2 10
2 9 13 13 16
3 10 14 7 10
4 8 14 1 4
5 16 20 11 14
6 13 20 5 6
7 23 27 13 14
```

```
8 21 25 8 10
9 22 26 0 3
```

Note que, neste caso, os retângulos coincidentemente não têm intersecção. Mas, no caso geral, pode haver intersecção entre os retângulos dados como entrada.

Saída

Seu programa deverá imprimir no dispositivo de saída padrão o valor numérico da área máxima encontrada. Para facilitar a depuração, outras informações podem ser escritas na saída padrão, da seguinte forma. Após o resultado, imprimir ponto-e-vírgula (";") seguido da informação extra desejada. O verificador de respostas corretas será instruído a ignorar tudo o que vier após o ponto-e-vírgula. Porém, em todos os casos, a saída padrão deverá ser constituída de uma única linha.

Uma sugestão interessante a colocar após o ";" é a lista dos identificadores dos retângulos admitidos no seu subconjunto de resposta. Esta informação pode ser útil na depuração.

Avaliação

Em princípio, a nota recebida será composta da seguinte forma:

- 70% nota de correção
- 10% nota de estilo
- 10% nota de comentários
- 10% nota de performace

A nota de correção será o quociente entre o número de testes abertos e fechados acertados e o número total de testes abertos e fechados.

A nota de estilo avaliará o bom estilo funcional do programador. Serão levados em conta aspectos como warnings, funcões não usadas ou desnecessárias, nomes de funções e variáveis, bom uso das funções pré-definidas, modularização, etc.

A nota de comentários avaliará a qualidade dos comentários dispostos no código. Este poderia ser um quesito do estilo, mas comentários são tão importantes que merecem um item de avaliação à parte.

A nota de performance avaliará a eficiência com que seu programa chega à resposta correta.

MC346 Home