Princípios de Programação

Projeto

Universidade de Lisboa
Faculdade de Ciências
Departamento de Informática
Licenciatura em Engenharia Informática

2019/2020

O Grupo de Alunos em Forma e Interessados em Programação Funcional (GAFIPF) estão a planear um *web site* sobre percursos para oferecer aos seus colegas do Departamento de Informática. Podem ficar com uma ideia do resultado esperado, visitando Kanto Adventures. A infraestrutura está quase pronta:

- Os percursos foram registados por elementos do GAFIPF usando um *GPS tracker*.
- Os dados relevantes dos percursos foram coleccionados em formato Comma-Separated Values (CSV).
- A interface *web* para exibir os vários percursos já está terminada.

Contudo, para carregar o *web site* com os vários percursos, é preciso (1) processar toda a informação que foi recolhida dos vários percursos, e (2) passar o resultado desse processamento ao *web site*, num formato que o *web site* o consiga processar. O objectivo deste trabalho é então, gerar um ficheiro JSON por percurso a partir de vários ficheiros (CSV) que contêm com toda a informação de cada percurso.

Gerar um ficheiro JSON a partir de ficheiros CSV

Escreva um programa que execute na linha de comandos. O programa lê da linha de comandos o caminho para *três* ficheiros: (1) um ficheiro CSV que descreve um percurso, (2) um ficheiro CSV com lista de pontos de interesse, e (3) um ficheiro JSON. Os dois primeiros ficheiros são de leitura e o terceiro ficheiro é de escrita.

Por exemplo, assumindo que os dois ficheiros de leitura se chamam `BairrosAntigosGPS.csv` e `LisboaPOI.csv` respectivamente, que o ficheiro de escrita é `Lisboa.json`, e que o ficheiro com a implementação do seu programa se chama `Main.hs`, abaixo um exemplo de interação com o programa:

```
$ ghc --make Main.hs
$ ./Main BairrosAntigosGPS.csv LisboaPOI.csv Lisboa.json
$ cat Lisboa.json
{
  "Categoria": "A",
  "Tempo␣total␣(m)": 175,
  "Ganho␣acumulado": 343,
  "Ganho␣acumulado␣por␣m": 0.036,
  "Pontos␣de␣Interesse": ["Mouraria","Miradouro␣Chao␣do␣
     loureiro",
    "Miradouro␣de␣Santa␣Luzia", "Feira␣da␣Ladra",
    "Alfama","Jardim","Miradouro","Fundador␣d␣Noticias",
    "Igreja␣Sao␣Roque","Eletrico","Bica",
    "Miradouro␣Sta␣Catarina","Largo␣Camoes","Fernando␣
       Pessoa",
    "Antigos␣Armazens␣do␣Chiado","Elevador␣Sta␣Justa",
    "Rossio","Monumento","D.␣Joao␣I","Praca␣da␣Figueira",
    "Arco␣Rua␣Augusta","Monumento","Cais␣das␣Colunas",
    "Terreiro␣do␣Paco","Se␣de␣Lisboa","Miradouro␣Sta␣
       Luzia",
    "Julio␣de␣Castilho","Miradouro␣Portas␣de␣Sol",
    "Zona␣Castelo␣Sao␣Jorge","Igreja␣Sao␣Vicente␣Fora",
    "Panteao␣Nacional","Lisboa"]
}
```

As subsecções abaixo descrevem o formato dos dois ficheiros de leitura e do ficheiro de escrita.

Especificação dos dois ficheiros de leitura

O primeiro ficheiro lista as coordenadas GPS de um determinado percurso. O formato de cada linha do ficheiro segue o seguinte formato:

- a primeira coluna contém a latitude (em graus)
- a segunda coluna contém a longitude (em graus)
- a terceira coluna contém a elevação (em metros)
- e a quarta coluna contém a hora no formato `HH:MM:SS` (Nota: o domínio de `HH` é $[00, 23]$, o domínio de `MM` é $[00, 59]$, e o domínio de `SS` é $[00, 59]$.)

Cada uma das colunas está separada por uma vírgula, de acordo com o formato CSV. O separador das casas decimais é o ponto. Por exemplo:

```
38.70755,-9.13649,1.097,17:36:45
38.70867,-9.13696,6.023,17:37:25
38.70960,-9.13732,9.025,17:38:05
38.71099,-9.13782,8.007,17:38:45
38.71238,-9.13834,14.063,17:39:25
38.71208,-9.13964,29.020,17:40:05
38.71102,-9.13970,27.013,17:40:45
38.71096,-9.13971,26.053,17:41:25
38.71065,-9.14208,40.053,17:42:05
38.71049,-9.14295,44.059,17:42:45
38.71032,-9.14384,47.053,17:43:25
```

O segundo ficheiro lista os pontos de interesse e cada linha do ficheiro segue o seguinte formato:

- a primeira coluna contém a latitude (em graus),
- a segunda coluna contém a longitude (em graus),
- a terceira coluna contém o nome de um ponto de interesse.

Por exemplo:

```
38.714,-9.1256,Panteao Nacional
38.713,-9.1238,St Apolonia
38.713,-9.1238,Museu Militar
38.7,-9.1,Lisboa
```

Especificação do ficheiro de escrita

Tempo total é dado em minutos e calculado pela diferença entre o primeiro e o último registo no ficheiro CSV das coordenadas, com arredondamento simétrico às unidades.

Ganho acumulado total corresponde aos metros totais subidos durante o percurso, com arredondamento simétrico às unidades. Atenção: As descidas não contam para o ganho acumulado. Por exemplo, uma subida de 2m, seguida de uma descida de 100m e de uma subida de 3m concretiza-se num ganho acumulado de 5m. A ideia é avaliar o esforço, não a diferença de altitudes.

Ganho acumulado por metro corresponde a quantos metros se subiu por cada metro que se andou na horizontal. Como dispomos apenas de coordenadas em graus, usamos a aproximação em que cada grau corresponde a 85km. Aqui utilizamos um arredondamento simétrico à terceira casa decimal.

Categoria A categoria é uma letra entre A e D, retirada de `http://www.kantoadventures.com/hiking-difficulty-scale.html`. A categoria A têm menos de 500m de ganho e menos de 10km de distância. A categoria B têm menos de 800m de ganho e menos de 12km. A categoria C têm menos de 1500m de ganho e um máximo de 15km. Tudo o resto tem categoria D.

Pontos de interesse contém todos os pontos de interesse por onde o percurso passa, apresentados em formato de lista de *strings*. Coloca-se a questão de saber se um percurso passa *num* ponto de interesse ou apenas *perto* do ponto. Para decidirmos se passa ou não usamos o conceito de *raio de vizinhança*. O raio de vizinhança de um dado ponto de interesse é definido pela resolução das coordenadas do ponto (assumimos que a resolução da latitude e da longitude é a mesma para cada ponto de interesse). Por exemplo, o raio de vizinhança de `Lisboa` é 10^{-1}, enquanto que o do `Museu Militar` é 10^{-3}. Assim, um percurso passa por um ponto de interesse se contiver um ponto que está dentro do raio de vizinhança do ponto de interesse.

Como testar o vosso programa?

Escreva **pelo menos três testes** para validar o programa desenvolvido:

1. O ganho acumulado é sempre positivo.
2. Se juntarmos um ponto de um percurso à lista dos pontos de interesse a considerar, aumentamos em uma unidade o comprimento da lista de pontos de interesse nesse percurso;
3. Pelo menos uma outra propriedade que considerar relevante.

Para executarem o vosso programa `Main` em modo de teste, podem utilizar a a opção `-t`, por exemplo:

```
$ ./Main -t
+++ OK, passed 100 tests; 135 discarded.
+++ OK, passed 100 tests; 122 discarded.
+++ OK, passed 100 tests; 43 discarded.
```

Notas

1. O seu trabalho é constituído pelo módulo `Main` e todos os outros módulos que achar convenientes. Respeite este nome.
2. Os trabalhos serão avaliados automaticamente.
3. Divida o seu trabalho em vários módulos.

4. Confine ações `IO` ao menor número de módulos possível.
5. Pode usar qualquer função constante no `Prelude` ou em qualquer outro módulo da biblioteca `Haskell`.
6. Não pode usar nenhum módulo que faça manipulação de CSV ou de JSON.
7. Não se esqueça de apresentar uma assinatura para cada função *top-level* que escrever.
8. Lembre-se que as boas práticas de programação Haskell apontam para a utilização de várias funções simples em lugar de uma função única mas complicada (o mesmo se aplica a módulos).

Entrega. Este é um trabalho de resolução em grupos de dois ou três alunos. Os trabalhos devem ser entregues no Moodle até às 23:55 do dia 18 de dezembro de 2019.
Deverá submeter um ZIP com todos os ficheiros necessários para a execução do `Main.hs` seja feita com sucesso. Os ficheiros CSV serão fornecidos por nós. O nome do zip deverá ser `fcXXXXX_fcYYYYY_fcZZZZZ.zip`, onde X, Y e Z são os algarismos dos número de aluno dos autores do trabalho.

Ética. Os trabalhos de todos os alunos serão comparados por uma aplicação computacional. Lembre-se: "Alunos detetados em situação de fraude ou plágio, plagiadores e plagiados, ficam reprovados à disciplina (sem prejuízo de ser acionado processo disciplinar concomitante)".