7ªaula prática

Árvores binárias de pesquisa

- Faça download do ficheiro ***aed2223_p07.zip*** da página e descomprima-o (contém a pasta ***lib*** ; a pasta ***Tests*** com os ficheiros ***dictionary.cpp, dictionary.h, funSetProblem.cpp, funSetProblem.h*** e ***tests.cpp*** ; e os ficheiros ***CMakeLists*** e ***main.cpp***)
- No CLion, abra um ***projeto***, selecionando a pasta que contém os ficheiros do ponto anterior.

1. Dicionários eletrónicos são ferramentas muito úteis. Pretende-se implementar um dicionário utilizando uma árvore de pesquisa binária (BST) onde as palavras se encontram ordenadas alfabeticamente. Considere que a árvore contém objetos da classe **WordMean**, e o dicionário está representado pela classe **Dictionary**.

```
class WordMean {
   string word;
   string meaning;
public:
   WordMean(string w, string m);
   //…
};
```

```
class Dictionary {
  set<WordMean> words;
public:
  void readFile(ifstream& fich);
  void print() const;
  string consult(string w1,
     WordMean& previous, WordMean& next) const;
  bool update(string w1, string m1);
};
```

1.1 Implemente o membro-função:

void Dictionary::readFile(ifstream& fich)

Esta função lê, a partir de um ficheiro (*), as palavras e o seu significado, e guarda essa informação na árvore *words*. Note que a árvore binária de pesquisa deve armazenar os elementos ordenados alfabeticamente pela palavra do dicionário. O ficheiro é composto por um número par de linhas, em que a primeira linha contém a palavra e a linha seguinte o seu significado:

```
gato
mamifero felino
morango
fruto
…
```

*** Nota:** Para aceder a ficheiros no seu programa (necessário ler o ficheiro *vets.txt*), pode:

a) especificar o caminho absoluto, ou

b) alterar "*Working directory*" no CLion (*Run -> Edit Configurations… -> Working directory*) para a pasta onde os ficheiros se encontram, ou

c) adicionar ao ficheiro *CMakeLists.txt* a diretiva

set(CMAKE_RUNTIME_OUTPUT_DIRECTORY "${CMAKE_CURRENT_SOURCE_DIR}/Tests")

neste caso, os ficheiros compilados e a ler/escrever são colocados na pasta *projeto*/Tests

1.2 Implemente o membro-função:

void Dictionary::print() const

Esta função imprime no monitor o conteúdo do dicionário, ordenado alfabeticamente por palavras, no formato seguinte:

```
palavra1
significado da palavra1
palavra2
significado da palavra2
…
```

nota: os testes unitários nunca falham

Complexidade temporal esperada: $O(n)$

1.3 Implemente o membro-função:

string Dictionary::consult(string w1, WordMean& previous, WordMean& next) const

Esta função retorna o significado da palavra *w1*. Se a palavra *w1* não existir no dicionário, a função retorna a string "*word not found*" e deve colocar em *next* e *previous* os objetos **WordMean** existentes no dicionário relativos às palavras imediatamente antes e imediatamente depois (ordem alfabética) da palavra *w1*, respetivamente. Se não existir palavra anterior e/ou palavra posterior, os objetos respetivos contém *word*="" e *meaning*="".

1.4 Implemente o membro-função:

bool Dictionary::update(string w1, string m1)

Esta função modifica o significado da palavra *w1* para um novo significado *m1*. Se a palavra *w1* existir no dicionário, o método retorna *true*, senão esta nova palavra com significado *m1* é adicionada ao dicionário e o método retorna *false*.

Complexidade temporal esperada: $O(n)$

2. Considere a classe **FunSetProblem** que possui apenas métodos estáticos. Implemente a função:

pair<int,int> FunSetProblem::pairSum(const vector<int>& values, int sum)

Dado um conjunto de valores inteiros não repetidos *values* e um valor *sum*, descubra se existe um par de valores cuja soma é *sum*. Se existir, a função deve retornar esse par, senão deve retornar (0,0).

Sugestão: use uma árvore binária de pesquisa para ir guardando os elementos do vetor. E enquanto percorre o vetor efetue também uma pesquisa à árvore para verificar se existe o "par" do elemento atual do vetor.

Complexidade temporal esperada: $O(n \times \log n)$

Exemplo de execução:

input: values = {7, 8, 12, 5, 2, 3, 5, 6} ; sum = 14

output: result = <8,6> ou <6,8>