3ªaula prática - Algoritmos de pesquisa. Documentação usando Doxygen

Instruções

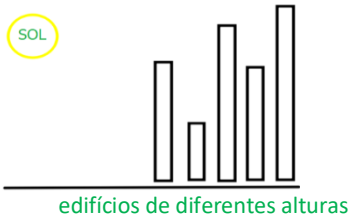
- Faça download do ficheiro ***aed2122_p03.zip*** da página da disciplina e descomprima-o (contém a pasta ***lib,*** a pasta ***Tests*** com os ficheiros ***funSearchProblem.h, funSearchProblem.cpp*** e ***tests.cpp***, e os ficheiros ***CMakeLists*** e ***main.cpp***)
- No CLion, abra um ***projeto***, selecionando a pasta que contém os ficheiros do ponto anterior.
- Efetuar "*Load CMake Project*" sobre o ficheiro *CMakeLists.txt*
- Execute o projeto (**Run**)

Nesta aula, deve:

1. Implementar o algoritmo para resolução do exercício enunciado
2. Definir os testes unitários apropriados para verificar a correção do algoritmo
3. Documentar o código, usando Doxygen**

Considere a classe **FunSearchProblem:**

```
class FunSearchProblem {
public:
    FunSearchProblem();
    static int facingSun(const vector<int> & values);
    static int squareR(int num);
    static int smallestMissingValue(const vector<int> & values);
    static int minPages(const vector<int> & values int numSt);
};
```



a) Implemente o membro-função:

int FunSearchProblem::facingSun(const vector<int> & values)

O vetor *values* representa a altura de edifícios em determinada rua, estando dispostos no vetor por número de porta. Supondo que o sol nasce no lado correspondente ao início da rua (ver figura), a função retorna o número de edifícios que verão o nascer do sol (isto é, o número de edifícios em que todos os que se encontram à sua esquerda são mais baixos). Implemente esta função usando apenas estruturas de dados lineares, devendo a solução apresentar complexidade temporal O(n).

b) Implemente o membro-função:

int FunSearchProblem::squareR(int num)

Esta função retorna a raiz quadrada de *num*. Se *num* não for um quadrado perfeito, então retorna o maior inteiro menor que raiz quadrada de *num* (isto é, a parte inteira do número que representa a raiz quadrada de *num*). Implemente esta função usando pesquisa binária e apenas estruturas de dados lineares.

c) (exercício extra) * Implemente o membro-função:

int FunSearchProblem::smallestMissingValue (const vector<int> & values)

O vetor *values* contém valores inteiros (positivos, negativos ou nulos). Esta função retorna o menor valor positivo não presente no vetor *values*. Se o vetor possui apenas valores negativos ou nulos, a função deve retornar *0*. Implemente esta função usando pesquisa sequencial e usando apenas estruturas de dados lineares.

d) (exercício extra) * Implemente o membro-função:

int FunSearchProblem::minPages(const vector<int> & values, int numSt)

O vetor *values* representa o número de páginas de um conjunto de livros dispostos numa prateleira. Os livros serão distribuídos por *numSt* estudantes para apoio à realização de um trabalho escolar. Só podem ser alocados a um estudante livros contíguos na prateleira. Cada estudante deve ter pelo menos um livro alocado a si.

A função deve encontrar a distribuição na qual o número máximo de páginas alocadas a um estudante é mínimo, e deve retornar esse valor mínimo. Se não existir uma distribuição válida, a função retorna -1. Implemente esta função usando pesquisa binária e usando apenas estruturas de dados lineares.

** Doxygen gera documentação a partir de código fonte. Passos a seguir (consultar também página moodle e Doxygen):

1. Instalar Doxygen.
2. Incluir no projeto a referência ao Doxygen, colocando essa informação em *"CMakeLists.txt"*.
   O texto apresentado a seguir serve como exemplo. Note a indicação (deve alterar para o que pretender) de:
   - ficheiro de configuração *Doxyfile* em "CMAKE_CURRENT_SOURCE_DIR}/docs/Doxyfile"

```
# Doxygen Build
find_package(Doxygen)
if(DOXYGEN_FOUND)
    set(BUILD_DOC_DIR "${CMAKE_SOURCE_DIR}/docs/output")
    if(NOT EXISTS "${BUILD_DOC_DIR}")
        file(MAKE_DIRECTORY "${BUILD_DOC_DIR}")
    endif()

    set(DOXYGEN_IN "${CMAKE_CURRENT_SOURCE_DIR}/docs/Doxyfile")
    set(DOXYGEN_OUT "${CMAKE_CURRENT_BINARY_DIR}/Doxyfile")
    configure_file("${DOXYGEN_IN}" "${DOXYGEN_OUT}" @ONLY)
```

```
    message("Doxygen build started")
    add_custom_target(Doxygen ALL
            COMMAND "${DOXYGEN_EXECUTABLE}" "${DOXYGEN_OUT}"
            WORKING_DIRECTORY "${CMAKE_CURRENT_BINARY_DIR}"
            COMMENT "Generating API documentation with Doxygen"
            VERBATIM)
else(DOXYGEN_FOUND)
    message("Doxygen needs to be installed to generate the documentation.")
endif(DOXYGEN_FOUND)
```

3. Criar o ficheiro de configuração *Doxyfile*, na pasta indicada em "*CMakeLists.txt*". Um exemplo deste ficheiro está na página moodle da UC.
4. Documentar o código fonte C++; lista de comandos pode ser consultada aqui.