Proposta de Trabalho

Project Proposal

Manipulação de vectores e estruturas dinâmicas usando a linguagem de programação C: caso de estudo dominó

Manipulation of arrays and dynamic data structures using the C programming language: domino case study

Algoritmos e Estruturas Dados I

Algorithms and Data Structures I

José Manuel Torres

jtorres@ufp.edu.pt

Linguagens de Programação I

Programming Languages I

Rui Silva Moreira

rmoreira@ufp.edu.pt

Outubro de 2016

October 2016

Universidade Fernando Pessoa

Faculdade de Ciência e Tecnologia

Faculty of Science and Technology

1 Descrição do Problema

O Dominó é um jogo tradicional constituído por várias peças rectangulares divididas ao meio. Cada uma das partes possui um número (sob a forma de pontos) entre 0 (ausência de pontos) e 6 (seis pontos). No total podem existir 28 peças correspondentes às várias combinações de números nas duas partes de cada peça.

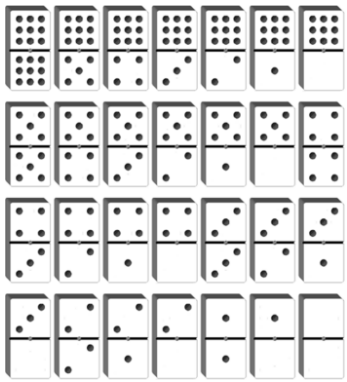

Figura 1: Conjunto total de peças de um jogo de dominó.

Na versão tradicional do dominó as peças são colocadas na mesa de jogo pelos jogadores de forma sequencial. Cada jogador coloca, à vez, uma das suas peças na mesa de jogo desde que esta encaixe numa das duas pontas da sequência de peças existente. Na sequência de peças visível todas as peças vizinhas possuem extremidades com o mesmo número de pontos.

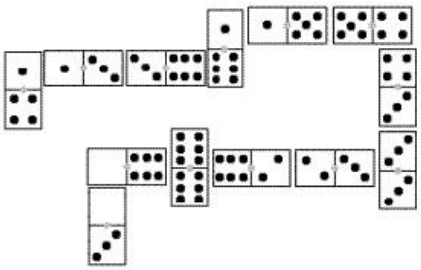

Figura 2: Exemplo de uma sequência possível de peças num jogo de dominó.

Neste trabalho pretende-se que implemente uma variação solitária do jogo do dominó tradicional, permitindo apenas um jogador (o computador) e incorporando as alterações que se descrevem a seguir:

1. Não existe baralho, apenas o jogo de mão inicial retirado do conjunto total de peças disponível. Durante o jogo não é possível "comprar" mais peças e acrescentá-las ao jogo de mão;
2. O jogo de mão pode ter, inicialmente, um número de peças diferente de 7 (inferior ou superior);
3. O jogador solitário pode jogar com mais do que um conjunto de peças, isto é, dois ou mais jogos de mão (mas todos os jogos de mão possuem, inicialmente, o mesmo número de peças). Neste caso o objetivo do jogador é obter a maior sequência possível de peças jogadas, retirando alternadamente uma peça de cada um dos seus jogos de mão.
4. O jogo termina quando não houver opções de encaixe disponíveis para o jogo de mão a ser utilizado nessa jogada.

Pretende-se implementar uma aplicação de software que permita fazer a gestão da variante de dominó proposta. A informação respeitante aos *logs* dos jogos deverá ser guardada e carregada a partir de ficheiros.

Os algoritmos desenvolvidos deverão ser usados na implementação do jogador computador para que este possa pesquisar qual a melhor jogada ou qual a melhor sequência de jogadas. Por exemplo, o jogador computador parte dum conjunto de n peças de dominó (jogo de mão único) e tenta criar uma sequência de encaixe que use todas as peças disponíveis (solução ótima). Nesta situação de jogo, e caso não consiga, o jogador computador deverá tentar usar o máximo número de peças disponíveis. Para o exemplo descrito a estratégia do jogador computador poderá seguir um algoritmo de pesquisa (e.g. *backtracking search*) que lhe permita descobrir qual a maior sequência de encaixe possível partindo de um conjunto inicial com *n* peças.

Neste projeto pretende-se que sejam especificados e desenvolvidos vários modelos de dados e algoritmos associados para a sua manipulação:

- Modelos de dados para representação das peças de dominó, usando tipos inteiros e/ou caracteres;
- Modelos de dados para representação de coleções de peças do tipo conjuntos de peças (e.g. jogo de mão ou baralho) onde a ordem das peças não é importante;

- Modelos de dados para representação de coleções de peças do tipo sequência (e.g. sequência de encaixe) onde a ordem das peças é importante, usando arrays e/ou estruturas dinâmicas;
- Modelos de dados para representação de coleções de sequências (coleção de coleções de peças de dominó);
- Modelos de dados para representação de informação sobre jogadores;
- Formatos de ficheiros para representar os modelos descritos acima;
- Vários algoritmos que poderão envolver ordenação e pesquisa.

2 Requisitos Funcionais

De seguida resumem-se os requisitos mínimos a cumprir neste projeto. Os requisitos deverão ter duas implementações (com exceção dos requisitos identificados):

i) Implementação baseada em **vectores e matrizes**;

ii) Implementação baseada em **apontadores** e **estruturas dinâmicas**.

Ambas as implementações deverão responder ou respeitar aos seguintes requisitos funcionais:

R1. Permitir ao utilizador armazenar e gerir um ou mais conjuntos de peças de dominó, i.e., inserir e/ou remover as peças iniciais dum jogo de mão usado no jogo; as peças também podem ser geradas de forma aleatória; em nenhum dos casos podem haver peças repetidas nos vários conjuntos de peças mão envolvidos;

R2. Dada uma representação de peças ou sequências de peças recorrendo a tipos inteiros, permitir gerar as mesmas peças ou sequência usando caracteres/strings, e vice-versa;

R3. Dadas várias sequências de encaixe com tamanhos variáveis (geradas por um algoritmo ou aleatoriamente) pretende-se permitir ordenar essas pesquisas por ordem de tamanho (comprimento da sequência de encaixe);

R4. Dadas várias sequências de encaixe com tamanhos variáveis (geradas por um algoritmo ou aleatoriamente) pretende-se encontrar padrões de encaixe nessas sequências, ou seja, encontrar as posições de sub-sequências pré-definidas de encaixe;

R5. Dada um sequência de encaixe com várias peças, permitir substituir um padrão de encaixe por outro padrão (os padrões podem ter tamanhos diferentes); deverá ser possível também substituir todas as ocorrências de um padrão por outro padrão, numa dada sequência;

R6. A partir de um conjunto de peças do jogador e de uma sequência de encaixe de peças inicial já colocada na mesa de jogo (esta sequência pode ser vazia), pretende-se determinar a sequência de encaixe final que utiliza todas as peças disponíveis; poderão existir uma ou mais sequências de encaixe com todas as peças; se não for possível encaixar todas as peças deve determinar-se a maior sequência de encaixe possível;

R7. Dados dois ou mais conjuntos de peças ou jogos de mão e de uma sequência de encaixe de peças inicial colocada na mesa de jogo (esta sequência pode ser vazia), pretende-se determinar se é possível encaixar todas as peças desses conjuntos de mão (retiradas à vez de cada mão do jogador) de forma a conseguir-se uma sequência válida; se não for possível encaixar todas as peças então deve determinar-se a maior sequência de encaixe possível;

R8. Permitir a manipulação das estruturas de dados através da entrada e saída de dados via ficheiros de texto;

R9. Permitir a manipulação das estruturas de dados através da entrada e saída de dados via ficheiros binários;

R10. Permitir a manipulação das estruturas de dados e das funcionalidades da aplicação através de uma estrutura de interação baseada em texto (menus); NB: este requisito aplica-se apenas a uma das implementações: i) ou ii).

3 Documentação

3.1 Anotações e comentários no código fonte

As estruturas de dados e os algoritmos definidos devem ser implementados em C com os comentários apropriados que facilitem a compreensão dos mesmos, inseridos no código fonte desenvolvido. Todas as funções devem estar anotadas em formato

doxygen[1] incluindo uma explicação dos algoritmos implementados e uma menção ao desempenho dos mesmos assim como dos testes efetuados/implementados. As principais estruturas de dados e variáveis devem também estar anotadas neste formato. Deverão usar o software doxygen para gerar a documentação com base nos comentários.

3.2 Relatório

Os alunos deverão entregar um ficheiro de texto no formato .dox[2] do doxygen, que descreva as funcionalidades/requisitos implementados, parcialmente implementados e não implementados. Devem mencionar sempre o número do requisito de acordo com a numeração utilizada neste documento de especificação:

1. Funcionalidades implementadas: devem descrever de forma resumida todas as funções desenvolvidas para assegurar os requisitos funcionais solicitados.
2. Funcionalidades não implementadas: nesta secção devem identificar de forma resumida as funcionalidades não implementadas; devem ainda apontar-se as justificações/dificuldades que impediram o seu desenvolvimento.

4 Submissão

A aplicação final (código) deve estar depurada de todos os erros de sintaxe e de acordo com os requisitos funcionais pedidos. Só serão considerados os programas que não contenham erros de sintaxe e implementem as funcionalidades pedidas. A documentação, em html ou pdf, juntamente com todo o código-fonte desenvolvido, deve ser submetida num ficheiro zip/rar (project.zip) na plataforma elearning.ufp.pt.

[1] A geração de anotações neste formato é normalmente suportada pelos IDEs ou plugins associados (http://www.doxygen.org).
[2] Ver o ficheiro "mainpage.dox" fornecido como exemplo.