Teoria dos Grafos

Universidade Federal de Itajubá - UNIFEI

Instituto de Engenharia de Sistemas e Tecnologia da Informação - IESTI

Guilherme Henriques Lasinskas – 2018002789

Gustavo de Alvarenga Guerra – 2018008684

João Victor de Moraes Santos Gomes – 2018008639

Nicholas Santana Santos – 2018002751

Rodrigo Sodré Coelho - 2018000363

1. Introdução

O jogo Ticket to Ride é um jogo de tabuleiro, tematicamente situado no início do século XX, especificamente em outubro de 1900, e seu tabuleiro representa o mapa dos Estados Unidos e todas as rodovias que ligam as cidades as outras.

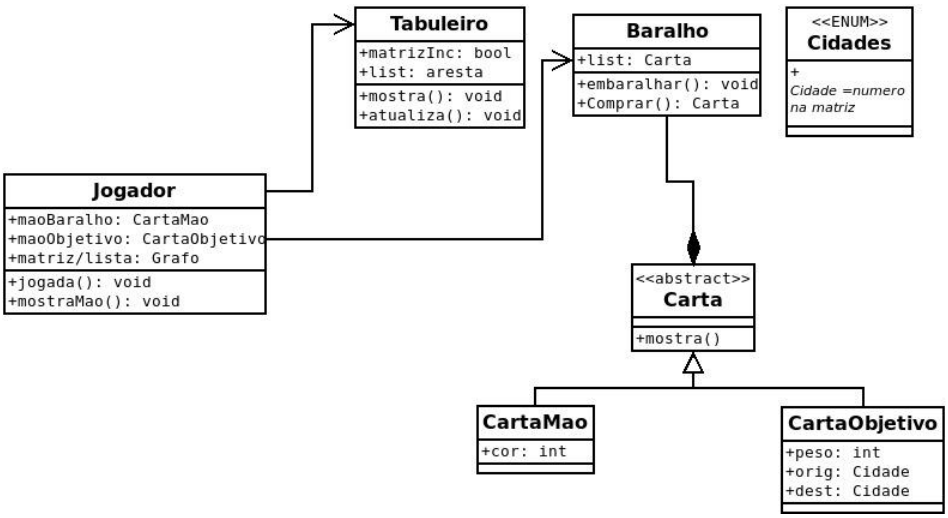
O jogo consiste em coletar cartas de vagões que por sua vez permitem a conquista de rotas de trem conectando duas cidades nos EUA. Quanto maior a rota, mais pontos são conquistados. Pontos adicionais são concedidos àqueles que conseguirem completar seus Bilhetes de Destino (conectando duas cidades distantes) e ao jogador que conseguir construir a maior ferrovia contínua.

O jogo possui grande relação com a disciplina de Teoria dos Grafos, uma vez que seu tabuleiro pode ser representado por um Grafo, e algoritmos como o de Dijkstra e o DFS podem ser aplicados no jogo, além de outros conceitos mais básicos como o de coloração, representação computacional de um grafo, ordem de um grafo, tamanho de um grafo e grafos ponderados**.**

2. O Planejamento Inicial

Inicialmente, foi entregue um projeto do jogo, que define classes e estruturas de dados que seriam usadas, e minimamente, alguns atributos e métodos dentro dessas classes.

O código do jogo havia sido pensado para possuir classes que representam, por exemplo, cartas de trem e objetivo, trens, o tabuleiro e os jogadores. A forma como havíamos pensado em representar cada classe está apresentada abaixo pelo diagrama em UML.



3. Metodologia e Implementação

Dentro do código do jogo, a parte que envolvia a função principal e toda sua implementação, foi dividida em dois tipos de matriz(usadas para representar os grafos computacionalmente): uma matriz de incidência que representa todo o tabuleiro e as ligações entre uma cidade a outra, e 5 matrizes de adjacência, uma para cada jogador, que por sua vez representam os trilhos que cada jogador possui dentro do tabuleiro. Essa parte está relacionada intrinsecamente relacionada à Teoria dos Grafos, na representação computacionais desses grafos através de matrizes.

Além do mais, utilizamos um DFS (Busca em Profundidade) para achar o maior caminho contínuo de cada jogador, também relacionando a disciplina de Teoria dos Grafos à implementação do jogo.

Já a implementação das classes ficou semelhante ao planejado anteriormente, apenas com algumas alterações que serão tratadas posteriormente. Todas as classes estão representadas abaixo.

Classe Baralho //baralho.h

```
#ifndef BARALHO_H
  #define BARALHO_H

  #include "cartatrem.h"

  class Baralho{
    public:
      /* Atributos */
      int primT,primO;// guarda o primeiro item da lista
      CartaTrem trens[110];
      CartaObjetivo objts[30];

      /* Contrutores e Destrutores */
      Baralho();
      Baralho(const Baralho& a);
      ~Baralho(){};

      /* Métodos */
      CartaTrem comprarTrem();
      CartaObjetivo comprarObjetivo();
      Baralho &operator=(const Baralho& a);
  };
#endif
```

Classe CartaObjetivo //cartaobjetivo.h

```
#ifndef CARTAOBJETIVO_H
  #define CARTAOBJETIVO_H

  class CartaObjetivo{
    public:
      /* Atributos */
      int peso, orig, dest;

      /* Contrutores e Destrutores */
      CartaObjetivo(){this->peso=-1;};
      CartaObjetivo(int p,int o, int d) {this->peso=p; this->orig=o;
                                                       this->dest=d;};
      ~CartaObjetivo() {};

      /* Métodos */
      friend ostream& operator<<(ostream&, const CartaObjetivo&);
      void mostraCarta();
  };

#endif
```

Classe CartaTrem//cartatrem.h

```
#ifndef CARTATREM_H
#define CARTATREM_H

  class CartaTrem{
    public:
      /* Atributos */
      int cor;

      /* Contrutores e Destrutores */
      CartaTrem(){cor=-1;};
      CartaTrem(int c){cor=c;};
      ~CartaTrem() {};

      /* Métodos */
      bool operator==(int c);
      friend ostream& operator<<(ostream&, const CartaTrem&);
  };

#endif
```

Classe Jogador//jogador.h

```
#ifndef JOG_ID
  #define JOG_ID

  #ifndef MAX
    #define MAX 45
  #endif
  #ifndef MAX_NO
    #define MAX_NO 45
  #endif

  #include "cartaobjetivo.h"
  #include "cartatrem.h"

  int Jogadores = 0;

  class Jogador{
  public:
    /* Atributos */
    int trem,objts;
    int maoTrem[9];
    CartaObjetivo maoObj[30];
    int ConquistasGrafo[MAX_NO][MAX_NO]; //adjacencias
    int contPeca;

    /* Contrutores e Destrutores */
    Jogador( int = 0 );
    ~Jogador(){};

    /* Métodos */
    void desejaDevolver();
    int buscaProfundidade(); //Retorna o maior caminho de um vértice
    int maiorCaminho(); //Retorna o maior dos caminhos
    void comprar(Baralho b,bool c); //COmpra carta do baralho

  };

#endif
```

Classe No//no.h

```h
#ifndef NO_H
#define NO_H
#include <string>

class No{
  public:
    /* Atributos */
    string nome[60];
    int peso;
    int cor;
    int dono;

    /* Contrutores e Destrutores */
    No(){};
    ~No(){};
};

#endif
```

Classe Tabuleiro//tabuleiro.h

```h
#ifndef TABULEIRO_H
  #define TABULEIRO_H
  #include "no.h"

  class Tabuleiro{
    public:
      /* Atributos */
      No MatrizInc [36][95]; //36 cidades e 95 arestas
      CartaTrem mesa[5];

      /* Contrutores e Destrutores */
      Tabuleiro(){};
      ~Tabuleiro(){};

      /* Métodos */
      void ConfiguraTabuleiro();
      void ConfigurarMesa(Baralho b, int lim);
      void AtualizaTabuleiro(int origem, int chegada, int aresta, int dono);
      CartaTrem compraMesa(Baralho, int);
  };

#endif
```

4. Diferenças entre o planejado e o executado

A primeira diferença entre as classes planejadas e as classes realmente implementadas é a diferenciação da Carta de Vagão e Carta de Objetivo em duas classes distintas, e não derivadas da mesma (Classe abstrata Carta), como planejado anteriormente.

Na classe Jogador, apenas foi adicionado uma fila para representar as cartas de objetivo que o jogador vai recebendo conforme o jogo é jogado.

Fora essas três classes, que tiveram seus tipos de método e estrutura modificados com maior intensidade, as outras classes tiveram sua ideia inicial mantida, apenas adicionando métodos e tipos de dados auxiliares, que foram inseridos devido a sua necessidade na implementação.

5. Conclusão

Ao compararmos nossa ideia inicial e nosso esboço inicial de como as classes seriam construídas e como nosso programa principal seria implementado, pode-se chegar à conclusão que nossa implementação seguiu o que foi planejado inicialmente, com apenas algumas alterações.

Foram utilizados conceitos como Orientação a Objeto e conceitos de Teoria dos Grafos, ao utilizarmos algoritmos conhecidos e conceitos básicos da disciplina. Ao tratar especificamente da disciplina de Grafos, utilizamos um algoritmo muito conhecido para a busca do maior caminho para cada jogador (Busca em Profundidade), além de utilizar os conceitos de representação computacional de um Grafo através de uma Matriz de Adjacência e Incidência, Grafo Ponderado e Coloração (em relação às arestas).