Trabalho 2 - Análise de Algoritmos

Entrega: 12 de Novembro

Esse trabalho consiste em resolver um problema de decisão sequencial utilizando grafos para modelar o espaço de estados.

O trabalho deve ser realizado **em dupla**, e pode ser feito em qualquer linguagem.

O que tem que ser entregue **no EAD**:

1. Relatório de 2-3 páginas com o conteúdo pedido abaixo
2. Código fonte
3. Executável **windows** (não precisa caso feito em Python)

Note que utilizarei sistema para controle de plágio de código, como o MOSS.[1]

Tarefa 1: Criação do grafo de espaço de estados

Considere um jogo que consiste de um tabuleiro reticulado 3x3 com 9 casas e 8 peças numeradas de 1 a 8. Ao longo do jogo sempre uma casa fica vazia e as demais contém uma única peça. Em cada jogada podemos mover para casa vazia uma peça que está localizada em uma casa adjacente (acima, abaixo, a direita, ou a esquerda) a ela. O objetivo do jogo e partir de uma dada configuração inicial das peças e chegar a uma configuração final dada. Para um exemplo veja `https://www.youtube.com/watch?v=op2Gsh0n3Fg`.

A Tarefa 1 consiste em implementar um procedimento que calcula o grafo não-direcionado do *espaço de estados desse jogo*. Nesse grafo temos: 1) um nó para cada configuração possível do tabuleiro; 2) arestas do tipo $(cfg1, cfg2)$ quando pudermos passar da configuração $cfg1$ para a configuração $cfg2$ em **um só movimento** do jogo.

Dica: Pode ser interessante usar tabelas hash para que o código rode em tempo hábil; você pode fazer isso, mesmo sendo que não vimos tabelas hash. Mais precisamente, pode ser útil ter uma tabela hash que dada configuração cfg, $hash[cfg]$ retorna o número do nó relativo a essa configuração (e vice-versa: um vetor C que dado o número u de um nó, $C[u]$ retorna a configuração correspondente a esse nó).

No relatório você deve reportar:

1. Quantos nós e aresta existem no grafo do espaço de estados que você construiu
2. Um exemplo de dois nós no grafo conectados por uma aresta
3. Um exemplo de dois nós no grafo que **não tem** um aresta entre eles

[1] `https://theory.stanford.edu/~aiken/moss`

Tarefa 2: Implementação de BFS e contagem de componentes conexos

Na segunda tarefa, você deve implementar um BFS que conta o número de componentes conexos do grafo construído na Tarefa 1.

No relatório você deve reportar:

1. O código principal da sua BFS (chamado nos slides de BFS(G,s), ou seja, o que contém o loop que visita os vizinhos dos nós)
2. Reporte quantos componentes conexos tem o grafo construído na Tarefa 1

Tarefa 3: Caminhos mais curto

Considere que o objetivo do jogo é chegar de uma configuração inicial dada à seguinte configuração final cfg^*:

Encontre qual é a configuração inicial **viável** mais difícil, ou seja, a que necessita o maior número de movimentos para se chegar a configuração cfg^* acima. Em linguagem de grafos: você deve considerar todas as configurações no componente conexo de cfg^*, e encontrar aquela cujo **caminho mais curto** à configuração cfg^* é **máximo**. Utilize BFS para realizar essa tarefa.

No relatório você deve reportar:

1. A configuração inicial viável que necessita o maior número de movimentos para se chegar a configuração cfg^*
2. O número de movimentos necessários para ir dessa configuração a cfg^*