Universidade Federal de Minas Gerais

Departamento de Ciência da Computação

Algoritmos I

Trabalho Prático 1

Proposta

05 de setembro de 2020

1 Introdução

Este trabalho é a implementação de uma resolução de um problema hipotético, decidir o ganhador do Jogo do Pulo.

O Jogo do Pulo consiste de um tabuleiro de n dimensões (nesse trabalho, só foi considerado n=2), em que alguns jogadores competem por chegarem primeiro à última posição do tabuleiro, que chamei de ”finish line”. O jogo começa com uma ordem e colocação predefinidos dos jogadores, e cada jogador tem o direito de se movimentar o número de casas associado ao ponto em que estiver. A partir daí, cada turno todos os jogadores jogam, sendo a ordem determinada, primeiro pela ordem inicial, e depois de acordo com o tamanho do passo anterior, ou seja, quem tiver dado o menor passo tem a vez.

É simples perceber que existem 3 condições de vitória, apresentadas abaixo em ordem decrescente de prioridade:

1. Chegar primeiro à finish line
2. Dar o menor penúltimo passo
3. Começar primeiro

2 Implementação

Assim, obtemos a missão de implementar um algoritmo que, eficientemente, nos diga, dadas certas condições iniciais, quem é o vencedor.

Primeiramente, as entradas dadas pelo arquivo parametrizado na chamada do executável são processadas dentro da classe ”init”, sendo então passadas para ”manager”, que as distribui de

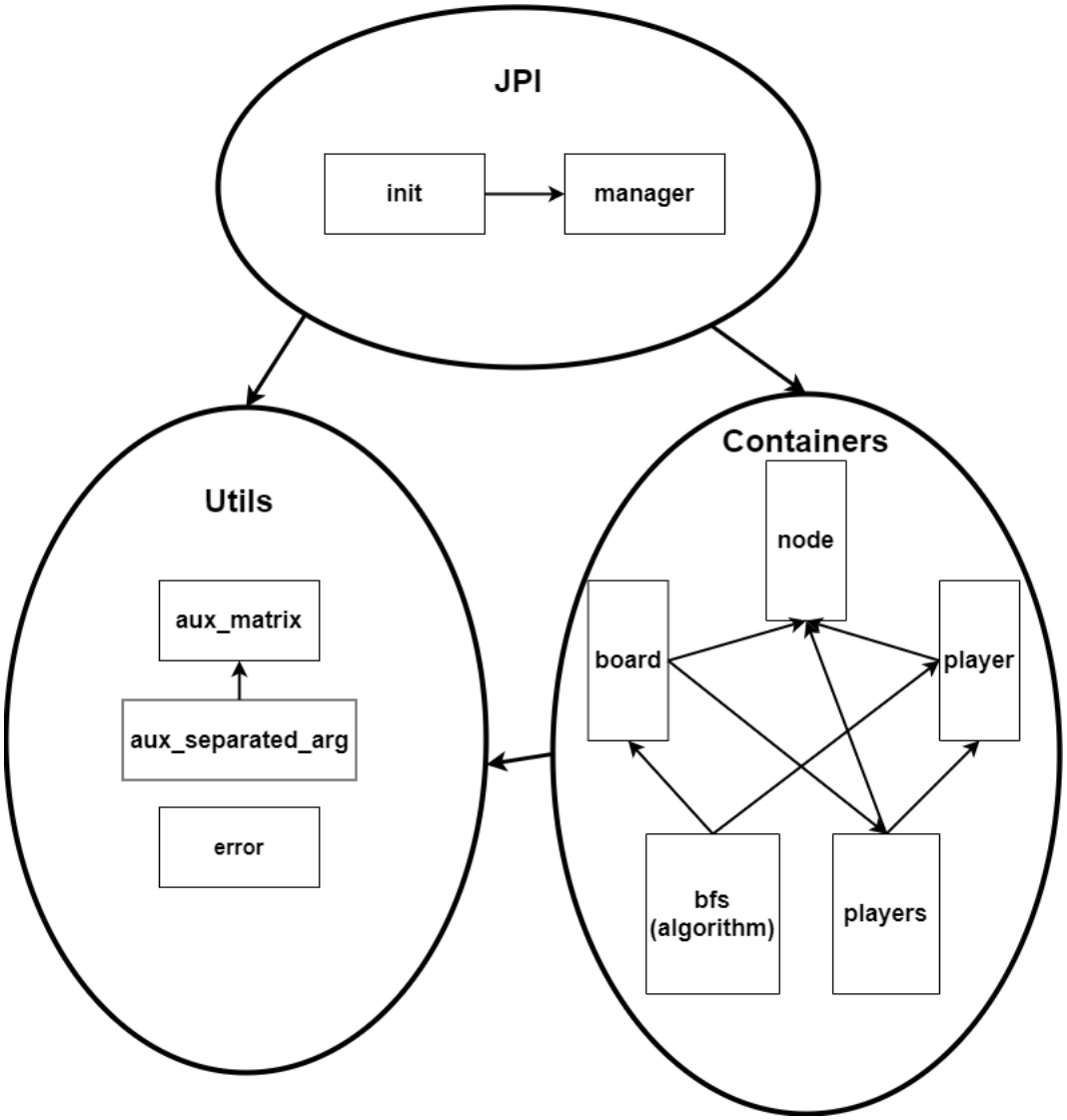

Figura 1: Diagrama de classes

acordo. Para auxiliar essa decodificação, também foram criadas as classes ”aux_matrix”e ”aux_separated_arg”no módulo ”Utils”.

Como sugerido na especificação do trabalho, o tabuleiro foi modelado como um grafo direcionado, em que cada vértice é uma posição. Nesse grafo, para um dado vértice, existe uma aresta de saída para cada posição alcançável, e uma aresta de entrada de cada posição que consegue chegar a ele com um passo. Cada jogador é simplesmente um desses vértices com um identificador associado, para que se possa distingui-los. (classes ”board”, ”node”, ”players”e ”player”)

O algoritmo está implementado no módulo ”Containers”. É basicamente um busca em largura em grafo direcionado. As modificações que adicionei seguem as seguintes observações:

- O primeiro passo é achar o conjunto de vértices que conseguem chegar à finish line dos restantes. Esses conjuntos são disjuntos.
- Como são disjuntos, uma vez que o conjunto com a finish line é encontrado, se um novo

vértice não pertencer a ele, já pode ser descartado como um possível caminho para a finish line.

- Enquanto não encontramos tal divisão, se algum vértice já houver sido explorado, não incluimos ele nos caminhos possíveis à finish line.
- Depois de encontrada, outro caminho que pode ser descartado é aquele que, no passado, já foi percorrido com mais êxito. Isto é, se chegamos em algum vértice que foi marcado com uma distância à origem menor que a atual, não vale a pena passar por ele.

3 Instruções de Compilação e Execução

Para compilar o programa, basta executar o comando make no diretório principal, que, utilizando o compilador local de C++11, deve gerar o arquivo ”tp1”. O comando ”make test”deve executar os testes locais e recompilar o programa se necessário.

4 Análise de Complexidade

Como mostrado na figura abaixo, gerada através dos arquivos ”src/test-measure.cpp”e ”src/test-measure.ipynb”, o algoritmo tem complexidade linear de ciclos de clock em função do tamanho da entrada. Isso condiz com a teoria, pois, por mais que haja muito ruído nessa medida, o peso real dos cálculos está na inicialização do grafo e execução do BFS, que, de fato, é O(m + n), sendo ”m”o número de arestas do grafo, e ”n”o número de vértices.

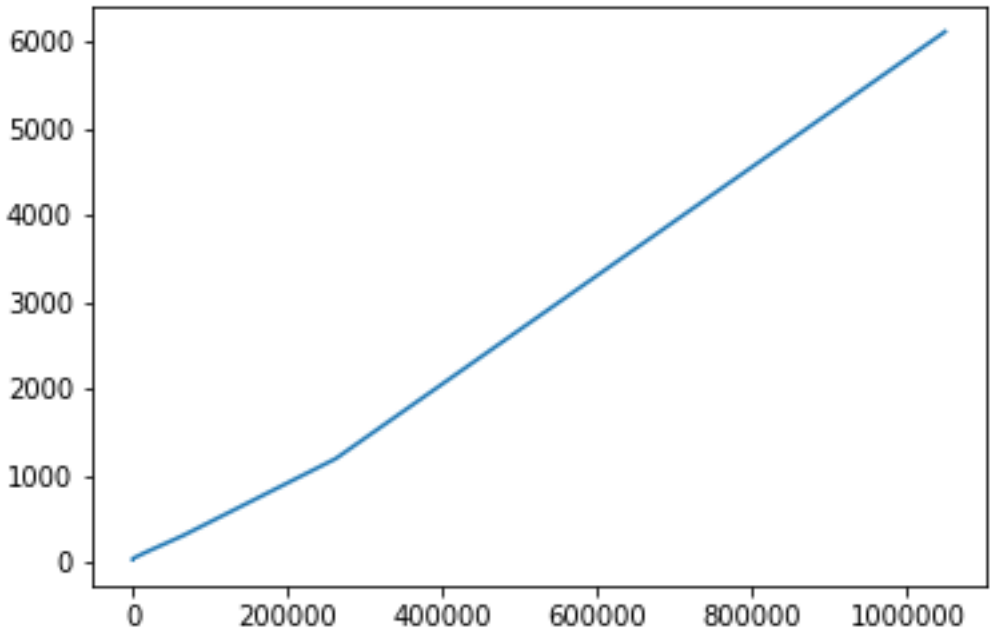

Figura 2: media dos ticks de clocks para 10 iterações, multiplicando o tamanho de 4 em 4

5 Conclusão

Este trabalho mostra o poder do algoritmo BFS. Inicialmente, eu pensava ser impossível resolver o problema dado em tempo viável, mas, através de uma modelagem em grafo, e observações cruciais (de que a maioria é, na verdade, inerente à estrutura de grafos e ao algoritmo BFS), isso se tornou possível.

Aprendi a respeitar essa estrutura de dados, e certamente considerarei utilizá-la ao encontrar outros desafios no futuro.

6 Bibliografia

https://ss64.com
http://www.cplusplus.com