Problema de Programação 1 - 2048

Student ID: 2016228735 Name: Afonso Matoso Magalhães
Student ID: 2017251509 Name: Leandro Pais

March 2021

1 Algorithm Description

O algoritmo base consiste numa função recursiva que para um dado tabuleiro gera todos os tabuleiros provenientes dos diferentes movimentos possíveis dentro do jogo. Assim, por exemplo, o primeiro nível gera quatro tabuleiros a partir do inicial. De seguida para cada um destes quatro são gerados novamente quatro. O algoritmo segue esta estrutura até que o nível de recursão seja igual ao número máximo de movimentos passado através da consola ou até que seja encontrada uma solução. Neste último caso existe uma variável persistente que é atualizada com o novo número de movimentos e passa depois a atuar como o novo número máximo de movimentos. O algoritmo irá esgotar todas e possibilidades e devolver este valor caso seja encontrado, ou *"no solution"* caso contrário.
A este algoritmo base aplicámos alguns conceitos apreendidos em aula. Para começar temos o pré-processamento que nos permite decidir se um dado tabuleiro tem solução ou não. Aqui aplicamos dois tipos, logo a quando da obtenção da matriz vamos somando todos os valores do tabuleiro e se

$$max(board) > \frac{soma}{2} \tag{1}$$

não existe solução e como tal a recursividade nem se quer é chamada. O segundo tipo, está presente nos swipes e consiste numa variável que nos dá o número de alterações que o movimento fez ao tabuleiro caso este valor seja zero paramos a recursão. Este mecanismo, permite-nos terminar a recursão cedo no caso de um tabuleiro estar "preso".
Por fim implementámos ainda uma verificação que permite cortar alguns ramos da árvore de recursão e consiste numa passagem à matriz para calcular o número de tiles ainda presentes no tabuleiro e se

$$\frac{tiles}{2} > (best - depth) \tag{2}$$

para tiles par, ou

$$\frac{tiles}{2} + 1 > (best - depth) \tag{3}$$

para tiles ímpar, a recursão termina.

2 Data Structures

No que toca às estruturas de dados utilizámos apenas uma matriz em que a cada posição corresponde um tile do tabuleiro de jogo. Fizemos ainda alguma experimentação com a conversão da matriz num array uni dimensional mas devido aos cálculos extra para cada posição no vetor que lhe são inerentes esta abordagem provou-se mais lenta. Assim, acabámos por manter apenas a matriz.

3 Corretcness

O total de pontos obtidos foi *180* com *"Time Limit Exceeded"* especulamos que a culpa deste resultado esteja inerente à nossa abordagem relativamente aos movimentos. Pelos cálculos efetuados apercebemos-nos de que o algoritmo implementado para eles está a correr com complexidade temporal $\mathcal{O}(n^2)$ que é de facto lento e pensamos que possa ser bastante otimizado.
Para além disto, tentámos também implementar heuristicas de procura de solução tais como a priorização dos swipes tendo em conta o número de merges que possam ser obtidos, a utilização de uma abordagem em que tentámos colocar os tiles com maior valor no canto inferior direito. No entanto, nenhuma das alternativas nos levou a mais do que *180* pontos, muito provavelmente por não terem sido implementadas da melhor forma pois estas estratégias estão aplicadas de forma bem sucedida em inteligências artificiais criadas para resolver o jogo original 2048.
Para concluir, pensamos que apesar da recursividade nos parecer bem conseguida temos o facto de que os swipes estão demasiado lento, a impedir-nos de alcançar os *200* pontos.

4 Algorithm Analysis

Como já foi referido, temos os swipes que correm em tempo quadrático ($\mathcal{O}(n^2)$). Quanto ao resto do código tentámos mantê-lo tão eficiente quanto possível utilizando a função memcpy sempre que possível para copiar o bloco inteiro de memória ao invés de fazer uma passagem pela matriz igualando cada um dos seus valores a uma outra, quando é detetado e efetuado um merge aumenta a potencia de dois em vez de estar a utilizar o $+=$ evitando assim um cálculo de acesso à memória, etc. No que toca ao algoritmo em si apresentamos a seguir os cálculos para a sua complexidade temporal.

$$T(n) = 4T(n) + C = 16(T(n) + 5C = 64T(n) + 21C \tag{4}$$

$$= 4^k T(n) + \frac{4^k - 1}{3} C \tag{5}$$

$$= T(n) \epsilon \mathcal{O}(4^{n+1}) \wedge C \epsilon \mathcal{O}(n^2) \tag{6}$$

No que toca à complexidade espacial, temos $\mathcal{O}(n)$, onde n é o tamanho da matriz, para cada chamada recursiva pois para cada nível de recursão apenas

fazemos uma cópia da matriz em questão. O mesmo é também verdade para os swipes pois para cada um mantemos um array adicional onde controlamos o número de merges efetuado em cada célula.

References

[1] Steve Mould's mathematical analysis of 2048 *https://www.youtube.com/watch?v=OO4tA5i7X9g&tw=481&ab_channel=SteveMould*.

[2] The Addictive Mathematics of the 2048 Tile Game *http://www.science4all.org/article/2048-game/*.