Relatório 1º projecto ASA 2022/2023

Grupo: AL039/TP11

Aluno(s): Gonçalo Rua (102604) e João Gouveia (102611)

Descrição do Problema e da Solução

A solução que encontrámos passa por encontrar o vértice mais à direita de um dado tabuleiro. Depois, encontramos o maior quadrado que seja possível inserir nesse canto e, assim, sabemos que os quadrados que podemos inserir nesse sítio são aqueles de medida entre a maior e 1. Seguidamente, geramos tabuleiros novos, cada um com um dos quadrados possíveis inseridos no seu canto mais à direita. O número de maneiras de ladrilhar o tabuleiro pedido vai ser a soma do número de maneiras de ladrilhar os novos tabuleiros criados. A condição de paragem da recursão é só ser possível inserir quadrados de lado 1 no tabuleiro ou termos um tabuleiro totalmente ladrilhado.
Se considerarmos a árvore de tabuleiros gerada por um dado tabuleiro, vão haver vários tabuleiros repetidos, existem várias formas distintas de ladrilhar parcialmente o tabuleiro incial e gerar um determinado tabuleiro específico. Para evitar calcular várias vezes as combinações destes tabuleiros repetidos, guardamos todos os tabuleiros cujo valor das possibilidades de ladrilhar já foi calculado numa hash table, aumentando assim a eficiência do nosso algoritmo.

Análise Teórica

A complexidade total do algoritmo é $O(X^2)$, sendo X o número de colunas da board. Isto porque a leitura da entrada é O(Y), mas o algoritmo roda a board (troca o Y com o X) sempre que Y >> X. Abaixo encontra-se análise mais promenorizada de cada função, com o seu respetivo pseudocódigo.

- Leitura de dados de entrada (read_input):

```
read_input()
    board ← new Board
    board.y ← Read input
    board.x ← Read input
    for i ← 0 to board.y - 1 do
        board.corners[i] ← Read input
    endfor
```

Simples leitura do input, com um ciclo que depende da altura (Y) da board, Θ(Y).

- Função inicial do problema (compute_board):

```
compute_board(board, table)
    if null_board(board) do
        return 0 // resultado do problema é zero
    end if
    if width < 0.75*height do
      spin_board(board)
    endif
    return get_combinations(board, table)
```

Função que faz uso da auxiliar null_board, O(Y), e da função auxiliar spin_board, O(Y), nos casos em que o tabuleiro é um retângulo com Y > X. Chama a função get_combinations, O(X^2), para obter o resultado do problema. Como rodamos o tabuleiro quando Y > X, então esta função é O(X^2).

- Função auxiliar para determinar se a board tem todos os cantos com valor zero (null_board):

```
null_board(board)
    for i ← 0 to board.y - 1 do
        if board.corners[i] != 0 do
            return false
        end if
    endfor
    return true
```

Função com um ciclo que percorre as linhas e termina se encontrar um canto com valor diferente de 0, logo O(Y). Os cantos são crescentes, logo, maior parte das vezes o tempo de execução desta função será bastante menor.

- Rodar a board (spin_board):

```
spin_board(board)
    corners ← new vector
    for j ← 0 to board.corners[0] - 1 do
        corners[j] ← board.y
    endfor
    for i ← 1 to board.y - 1 do
        if board.corners[i] == board.x do
            for j ← j to board.x - 1
                corners[j] ← board.y - i
            endfor
            Exit for
        else
            while i < board.y and board.corners[i] == board.corners[i-1] do
                i ← i+1
            endwhile
            for j ← j to board.corners[i] - 1 do
                corners[j] ← board.y - i
            endfor
        endif
    endfor
    for j ← j to board.x – 1 do
            corners[j] ← 0
    endfor
    board.x ↔ board.y
    board.corners ← corners
```

O primeiro loop tem no máximo X iterações, pelo que é O(X). O segundo loop terá X ou

Y iterações, dependendo do tabuleiro a tratar. No entanto, sabemos que esta função apenas é chamada quando Y > X, logo, o segundo loop é O(Y). Quanto ao último loop, tal como o primeiro, tem no máximo X iterações e, por conseguinte, é O(X). Atendendo ao facto desta função só ser chamada quando Y > X, podemos concluir que é O(Y).

- Obtém o hash id de uma board (hash):

```
hash(board)
   hash ← 0
   for c in board.corners do
      hash ← 17*hash + hash_number(i)
   endfor
```

Tem de percorrer todos os corners (que correspondem à altura), logo O(Y).

- hash_number: O(1)

- Função principal que obtém o resultado (get_combinations):

```
get_combinations(board, table)
   if finished_board(board) do
         return 1
   endif
   if lookup_board(board) do // if board is in table
       return table[board].combinations
   endif
   answer ← 0
   children ← new vector
   place_rightmost_tile(children, board)
   for b in children do
       answer ← answer + get_combinations(b, table)
   endfor
   insert_board(answer, board, table)
   return answer
```

finished_board é O(Y), lookup_board e insert_board são O(1) e place_rightmost_tile é $O(X^2)$. children tem no máximo X elementos, uma vez que, devido á função spin_board, X > Y (caso isto não se verifique é porque a diferença é mínima e podemos afirmar, aproximadamente, X = Y). Assim, temos $T(X) = XT(X^2 - b) + O(X^2)$, pois, mais uma vez, X > Y. Logo, assumindo um b insignificante, temos que esta função é $O(X^3)$ no caso em que não se repetem casos. Considerando o uso da hashtable para casos repetidos, verificamos que, em média, 1/3 dos casos já lá se encontram, concluindo que esta função é $O(X^2)$.

Relatório 1º projecto ASA 2022/2023

Grupo: AL039/TP11

Aluno(s): Gonçalo Rua (102604) e João Gouveia (102611)

- Pesquisa o indíce de uma board na hashtable (lookup_board):

```
lookup_board(board, table)
   answer ← -1
   while id != table.end do
      if table[id].board == board do
         answer ← id
         Exit while
      endif
      id ← hash_number(id)
   endwhile
   return answer
```

Pesquisa o indíce de uma board na hashtable, levando em conta o método de tratamento de colisões (rehashing). É Θ(n), sendo *n* o número de elementos da table, mas como colisões são raras e muitas vezes não necessitam mais do que duas iterações do while, consideramos que esta função é O(1).

- Adiciona uma board à hashtable (insert_board):

```
insert_board(answer, board, table)
   cell ← new cell(answer, board)
   while id != table.end do
      id ← hash_number(id)
   endwhile
   table[id] = cell
```

Adiciona uma board, com o seu resultado, à hashtable levando em conta o método de tratamento de colisões (rehashing). É Θ(n), sendo *n* o número de elementos da table, mas como colisões são raras e muitas vezes não necessitam mais do que duas iterações do while, consideramos que esta função é O(1).

- Verifica se chegámos ao fim do cálculo da board passada como argumento (pois o maior quadrado já só pode ter tamanho 1) (finished_board):

```
finished_board(board)
   for i ← 0 to board.y do
      if board.corners[i] > 1 do
          return false
      endif
   endfor
   return true
```

Função com um ciclo que percorre as linhas e termina se encontrar um canto com valor superior a 1, logo O(Y).

- Verifica se duas boards são iguais (equal_boards):

Relatório 1º projecto ASA 2022/2023

Grupo: AL039/TP11

Aluno(s): Gonçalo Rua (102604) e João Gouveia (102611)

```
equal_boards(board1, board2)
    if board1.y != board2.y or board1.x != board2.x do
        return false
    endif
    for i ← 0 to board1.y do
        if board.corners[i] != board2.corners[i]
            return false
        endif
    endfor
    return true
```

No pior caso as boards são iguais e a função terá de percorrer todos os corners, que corresponde ao número de linhas da board, logo O(Y).

- place_rightmost_tile:

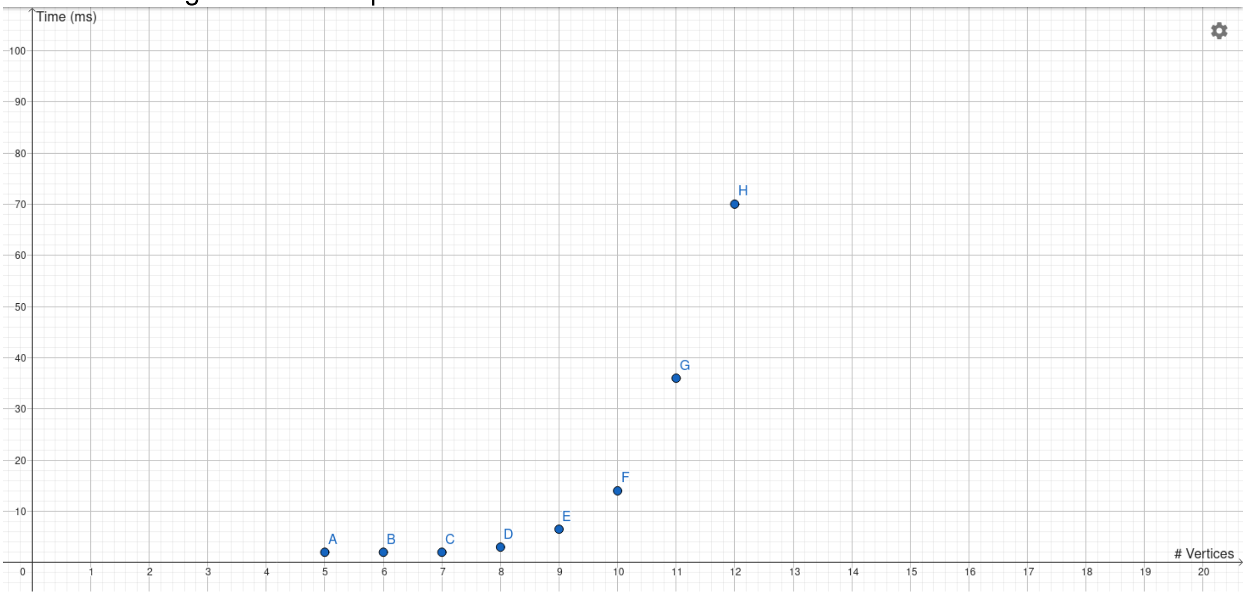
```
place_rightmost_tile(result, board)
    max ← 0
    for i ← 0 to board.y - 1 do
        if board.corners[i] > board.corners[max] do
            max ← i
        endif
        if board.corners[max] == board.x do
            Exit for
        endif
    end for
    tile_size ← 0
    if board.y – max >= board.corners[max] and board.corners[max] > tile_size do
        tile_size ← board.corners[max]
    endif
    if board.coners[max] >= board.y – max and board.y – max > tile_size
        tile_size ← board.y – max
    endif
    for i ← 0 to tile_size - 1 do
        if board.corners[max+i] < board.corners[max] do
            tile_size = i
            Exit for
        endif
    endfor
    for tile_size ← tile_size to 0 (descending) do
        new_corners ← new corners with same content as board.corners
        new_board ← new board with same x and y as board and new_corners
        for i ← 0 to tile_size - 1 do
            new_board.corners[max+i] ← new_board.corners[max+i] – tile_size
        endfor
        Add new_board to the end of result
    endfor
```

O primeiro loop é O(Y) e o segundo depende do tamanho do ladrilho máximo, pelo que é O(X) ou O(Y). O terceiro for loop também depende do tamanho do ladrilho máximo, que é $O(X^2)$ ou $O(Y^2)$ pois é o (somatório de i=0 até tile_size – 1) do (somatório de j=0 até i-1). new_corners terá no máximo X ou Y elementos. Sabemos que regra geral, devido á função spin_board, X > Y (caso isto não se verifique é porque a diferença é mínima e podemos afirmar, aproximadamente, X = Y), com vista a isto, conclui-se que esta função é $O(X^2)$.

Avaliação Experimental dos Resultados

Para a avaliação experimental, foram gerados testes progressivamente maiores. A temporização da duração da resolução de cada teste foi feita com recurso ao comando “time”.

As instâncias geradas são apresentadas no fim do documento.

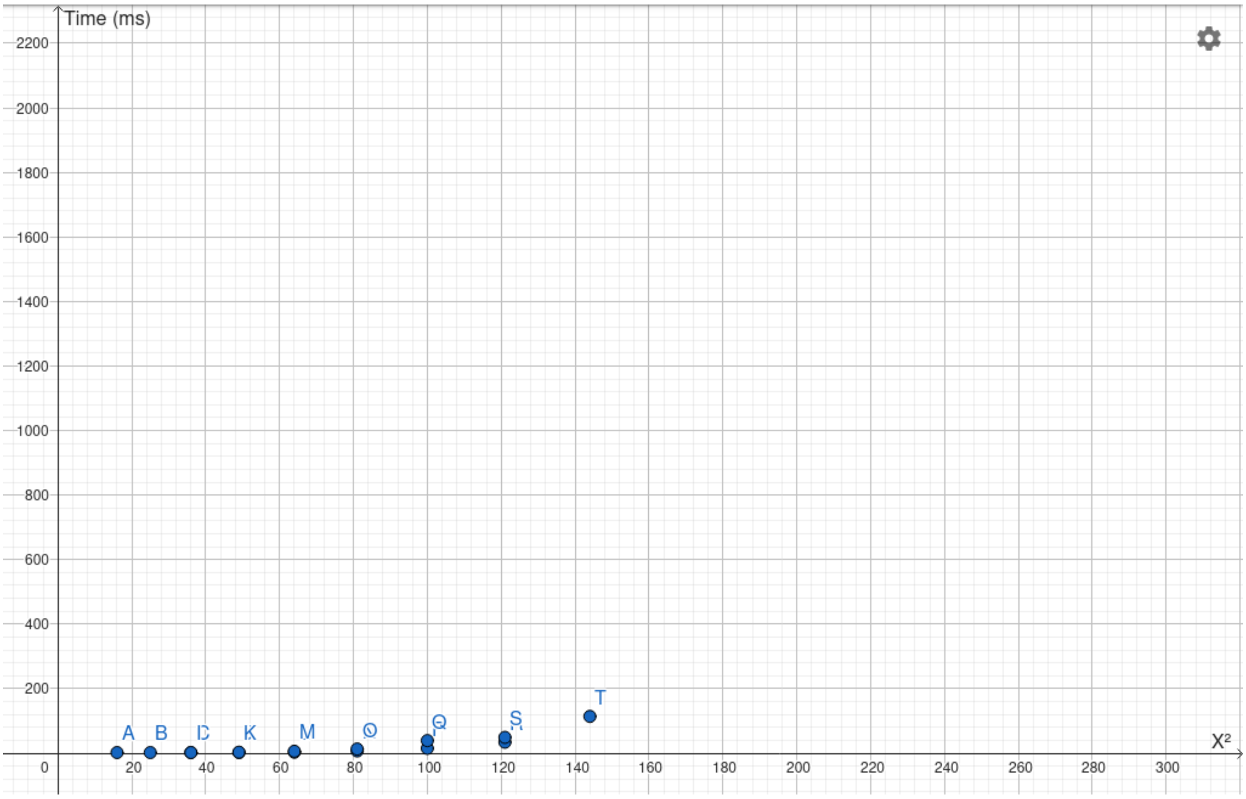


Logicamente, o gráfico não é linear quando o eixo X varia de acordo com o número de vértices. Trocando a variação deste eixo para uma coerente com o que foi proposto teoricamente:

Relatório 1º projecto ASA 2022/2023

Grupo: AL039/TP11

Aluno(s): Gonçalo Rua (102604) e João Gouveia (102611)



Verificamos rapidamente que se trata de um gráfico linear, que corrobra a nossa análise teórica.

Abaixo estão os testes utilizados:

10x10

10

10

10

10

10

10

10

10

10

10

10

10
10x9

10

9

9

9

9

9

9

9

9

9

9

9

9x9

9

9

9

9

9

9

9

9

9

9

9

9x8

Relatório 1º projecto ASA 2022/2023

Grupo: AL039/TP11

Aluno(s): Gonçalo Rua (102604) e João Gouveia (102611)

9

8

8

8

8

8

8

8

8

8

8

8x8

8

8

8

8

8

8

8

8

8

8

8x7

8

7

Relatório 1º projecto ASA 2022/2023

Grupo: AL039/TP11

Aluno(s): Gonçalo Rua (102604) e João Gouveia (102611)

7

7

7

7

7

7

7

7

7x7

7

7

7

7

7

7

7

7

7

7x6

7

6

6

6

6

6

6

6

Grupo: AL039/TP11

Aluno(s): Gonçalo Rua (102604) e João Gouveia (102611)

6

6x6

6

6

6

6

6

6

6

6

6x5

6

5

5

5

5

5

5

5

5x5

5

5

5

5

5

5

5

5x4

5

Relatório 1º projecto ASA 2022/2023

Grupo: AL039/TP11

Aluno(s): Gonçalo Rua (102604) e João Gouveia (102611)

4

4

4

4

4

4