

Relatório 1º projecto ASA 2022/2023

Grupo: TP053

Aluno(s): Rui Amardal (103155)

Descrição do Problema e da Solução

Para descobrir todas as maneiras de ladrilhar uma área, recorreu-se à programação dinâmica de forma a dividir cada situação em vários subproblemas cada vez mais simples, até chegar a um caso base, cuja solução se conhece.

Na solução implementada, cada área é representada por um vetor com n entradas, cada uma representando o comprimento da linha com esse índice.

Para resolver um problema, encontramos o ponto mais à direita (coluna mais comprida) com índice mais alto e obtemos vários subproblemas retirando quadrados com “raíz” nesse ponto, um subproblema para cada tamanho de quadrado que é possível retirar nesse ponto (nunca terá lado maior que n). Cada subproblema será então equivalente ao problema “pai” com um quadrado (de lado ≥ 1) a menos, que lhe foi retirado do ponto mais à direita da área.

A solução do problema original será então igual à soma das soluções dos seus subproblemas. Como vamos removendo quadrados às áreas, estas vão ficando mais reduzidas até que se chega a uma situação em que a coluna máxima tem tamanho < 1 , cujo número de soluções é conhecido (tem apenas uma solução que corresponde a ser preenchida por quadrados de lado 1).

À medida que se resolvem as várias situações, vão-se armazenando as soluções de problemas individuais caso o programa necessite de aceder a esse valor mais tarde durante a sua execução.

Análise Teórica

Seja N o lado de uma área quadrada (em que $n = m$ e cada linha tem comprimento máximo).

Durante a execução do programa temos as seguintes fases:

- Leitura dos dados de entrada: simples leitura do input, com ciclo(s) a depender linearmente de N . Logo, $O(N)$.
- Verificação dos dados lidos (para identificar o caso em que todas as colunas têm comprimento 0) em tempo constante. Logo, $O(1)$.
- Chamada de função recursiva:
 - Verificação se o problema já foi resolvido durante a execução do programa (verificar se uma certa key está presente numa hashtable). Em média realizar-se-á em tempo constante, mas no pior caso é $O(N)$.
 - Iteração sobre vetor problema para encontrar coluna mais comprida que depende linearmente de N . Logo $O(N)$.
 - Verificação se a coluna maior obtida é maior que 1 (para identificar o caso base) em tempo constante. Logo, $O(1)$.

Relatório 1º projecto ASA 2022/2023

Grupo: TP053

Aluno(s): Rui Amaral (103155)

- Extração dos subproblemas através de dois ciclos que dependem linearmente de N . Logo $O(N^2)$.
- Invocação recursiva da função dentro de um ciclo que depende linearmente de N , logo, esta será invocada $O(N)$ vezes.
- A cada chamada da função recursiva, os vetores problema vão tendo cada vez menor área, supomos então que a cada chamada da função, N diminui. Ou seja, teremos aproximadamente: $T(N) = N \cdot T(N-1) + N^2$.
- Apresentação do resultado em tempo constante. Logo, $O(1)$.

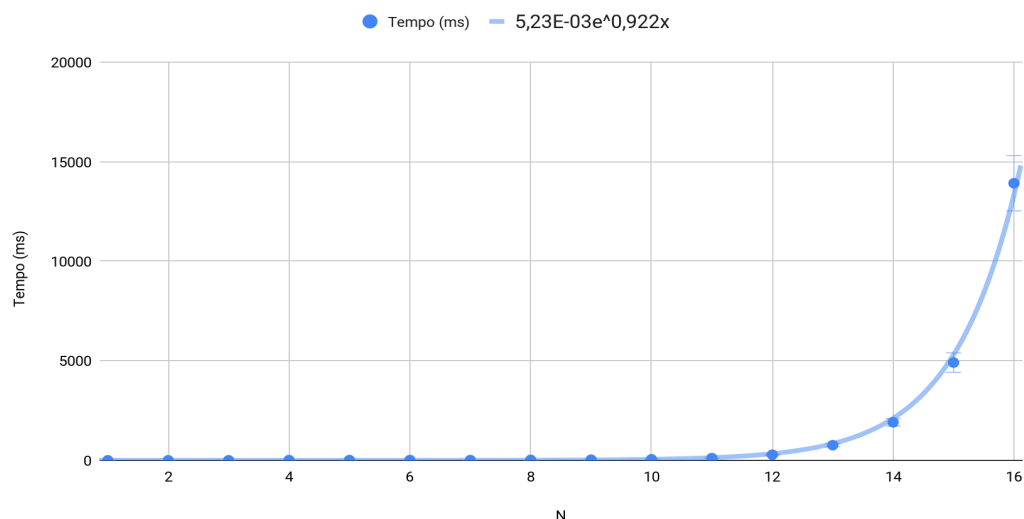
Complexidade global da solução: $O(N + T(N))$

Avaliação Experimental dos Resultados

Ao analisar o comportamento do programa, podemos determinar a complexidade temporal na prática.

Foi feito o registo do tempo que o programa demora a resolver problemas quadrados (em que $n = m$ e cada linha tem comprimento máximo). Seja N a medida do lado de cada problema. Ao fazer este registo para 16 valores de N (de 1 a 16), obtemos o seguinte gráfico:

Tempo (ms) em comparação com N



Com este conjunto de pontos, obtemos a equação da linha: $5,23E-3e^{0,922N}$.

Esta experiência permite determinar a complexidade temporal prática da solução que é então definida por: $O(e^N)$.