**Descrição do Problema e da Solução**

A nossa solução foi remover ladrilhos sempre em formato de quadrado com diferentes tamanhos (ex: 1x1, 2x2 , 3x3…), chamando sempre o algoritmo para resolver os subproblemas que surgiram da remoção desses quadrados. Para tal funcionar os quadrados não podem ser removidos arbitrariamente, estes são então sempre removidos da entrada mais à direita da linha com mais colunas e no caso de haver mais que uma linha com o maior número de colunas é escolhida sempre a linha mais a baixo possível, por exemplo: numa matriz 3x3 cujo os ladrilhos são [2,2,3], a linha com mais colunas é a última, portanto íriamos remover todos os quadrados de tamanhos diferentes possíveis cujo canto coincide com a entrada (3, 3) (entrada mais à direita da linha).

Portanto o nosso algoritmo funciona da seguinte forma: primeiro procuramos a linha com mais colunas, depois é calculado o maior quadrado (sxs) que é possível formar com o canto inferior direito a começar na entrada mais à diretida dessa linha, são então formados s subproblemas diferentes (um em que foi removido um quadrado 1x1, outro 2x2…, ate sxs) e este algoritmo é aplicado a todos estes subproblemas e a todos os subproblemas que surjam destes. A resposta ao problema é então a quantidade de vezes que se chega ao caso em que já não há mais ladrilhos a ser retirados.

**Análise Teórica**

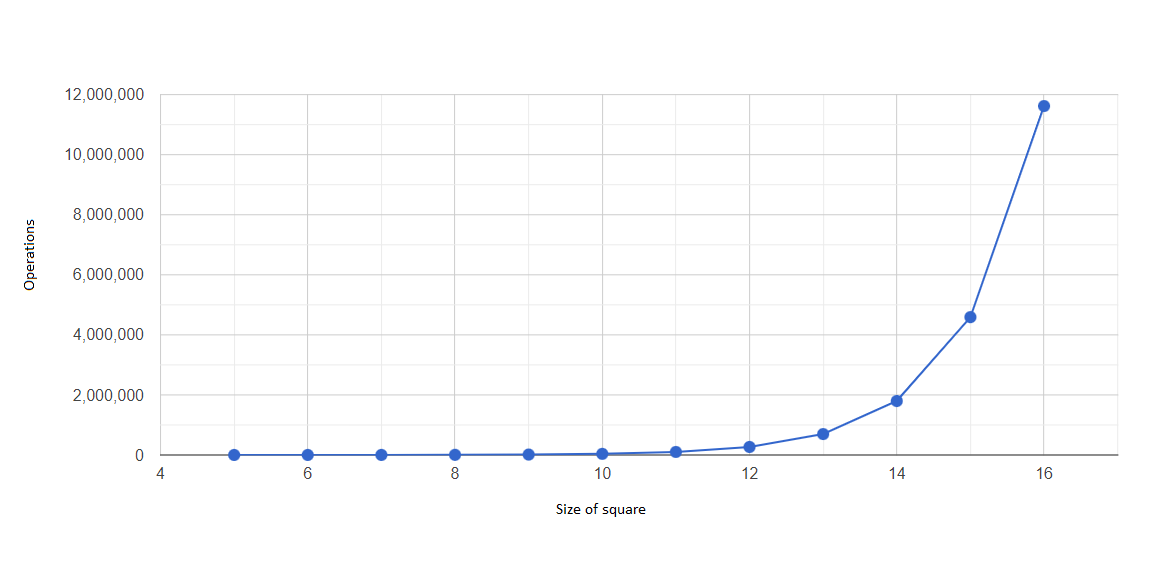
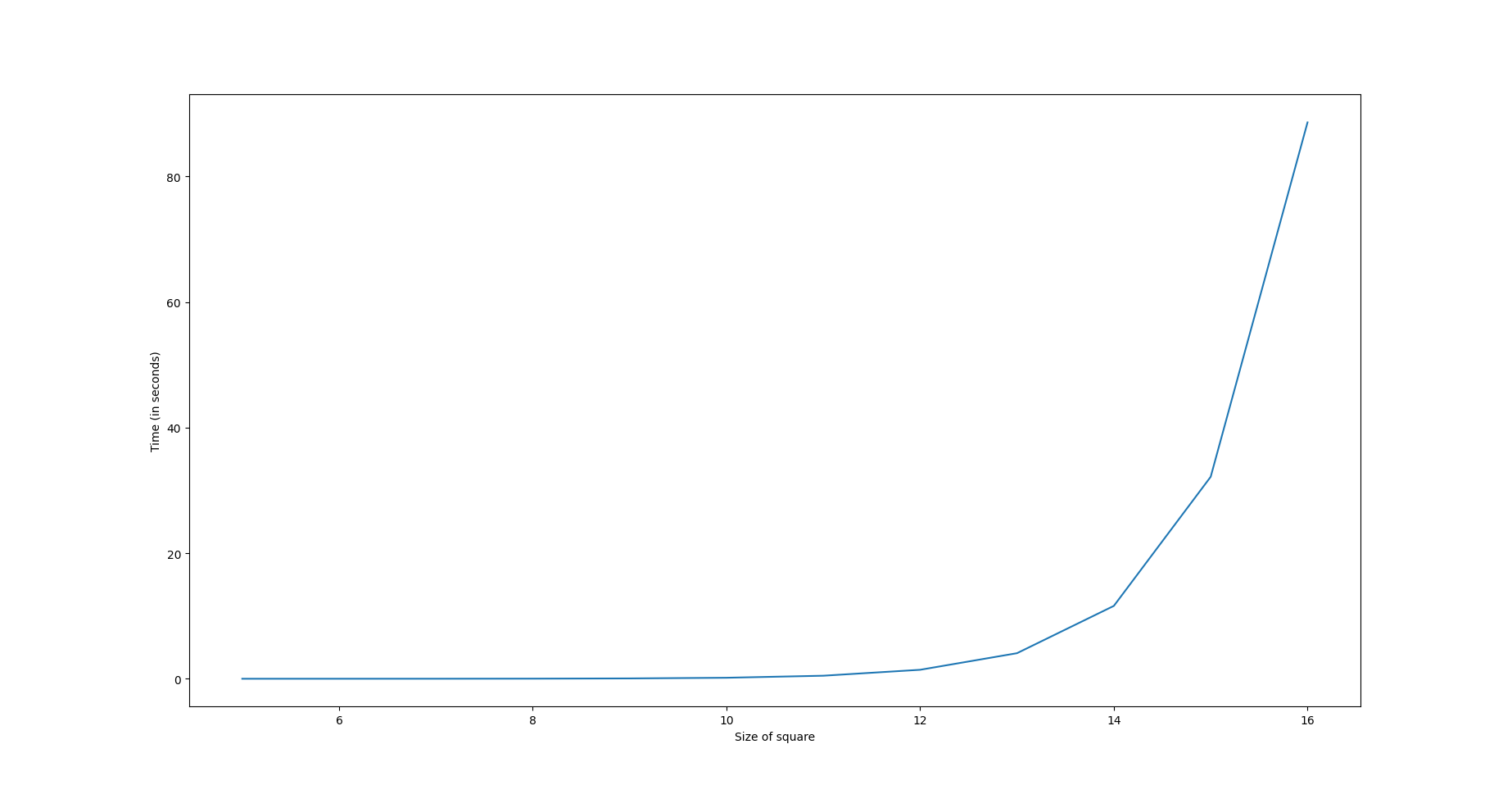
n = número de linhas, m = número de colunas

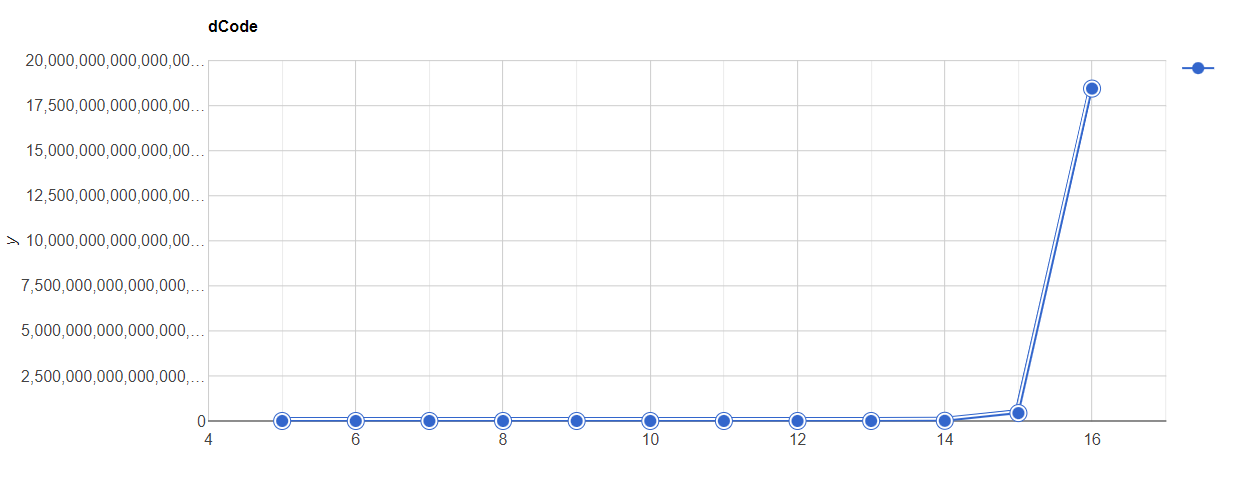
* Leitura dos dados de entrada: simples leitura do input, com um ciclo a depender linearmente de n – O(n).
* Verificação de que existem ladrilhos a ser removidos com um ciclo a depender n – O(n).
* Aplicação do algoritmo recursivo:
  + Procura pelo caso já resolvido (memoization) – O(1).
  + Procura pela linha com mais colunas, com um ciclo a depender linearmente de n – O(n).
  + Calculo do maior quadrado possível, com um ciclo a depender linearmente de n – O(n)
  + Iteração sobre todos os quadrados possíveis - O(nm):
    - Remocao do quadrado – O(n)
    - Chamada recursiva: o algoritmo em si utiliza a “procura pela linha com mais colunas” e o “calculo do maior quadrado” ambos O(n), sendo este chamado recursivamente e a profundidade da recursão dependendo de m - O(nm)

**Complexidade Global: O(nm)**

**Avaliação Experimental dos Resultados**

Apresentamos de seguida dois gráficos (lado do quadrado, número de operações) e (lado do quadrado, tempo), para ambos os gráficos foram usados quadrados de 5x5 até 16x16 (logo 12 instâncias).



Como foi dito anteriormente a nossa complexidade teórica é O(nm), o que não é demonstrado pelo gráfico prático das operações (este pode ser comparado com o gráfico teorico de uma nm a baixo). Esta diferença entre resultados pode ser justificada pela utilização de memoization que torna o nosso algoritmo muito mais eficiente em termos de tempo e número de operações.