Relatório 1º projecto ASA 2022/2023

Grupo: TP049

Alunos: Bernardo Galante (102423) e Henrique Machado (103266)

Descrição do Problema e da Solução

Este projeto visa a criar uma solução para o problema: quantas configurações de quadrados de qualquer dimensão inteira existem para ladrilhar uma área limitada de um retângulo com dimensões n por m.

De modo a resolver este problema optámos por dividir o retângulo inicial em vários sub-problemas, de modo a facilitar a sua resolução. A divisão do problema inicial é feito de forma recursiva, ou seja, situamo-nos no quadrado 1x1 da maior coluna da área a ladrilhar e calculamos o maior quadrado possível a retirar. Com o quadrado maior (**max***max**) calculado, dividimos o problema em **max** problemas onde é retirado ao retângulo anterior um quadrado de 1*1 até **max***max** e voltamos a aplicar a recursão aos **max** problemas criados até chegarmos a um retângulo vazio. O número de retângulos vazios encontrados é a solução do problema. Implementámos um hashmap de forma a aplicar o conceito de **memoization** e tornar a resolução mais eficiente. Assim, sempre que nos deparamos com um sub-problema já resolvido, evitamos calcular a sua solução outra vez e obtemos logo o seu resultado.

Análise Teórica

- Leitura dos dados de entrada: lê os valores da dimensão do retângulo (n-linhas e m-colunas) e depois lê, com um ciclo a depender linearmente do valor n, valores correspondentes ao número de colunas de cada linha. Logo Θ(n).

- Processamento dos dados de entrada (função **create_board()**) para criar um **board** (struct contendo as dimensões e índice da maior coluna, e um vetor de inteiros com os valores de cada linha). Tem um ciclo a depender linearmente do valor de n (número de linhas). Logo Θ(n).

- Verificação do estado do retângulo (função **isEmpty(board b)**) iterando por cada valor da linha e verificando se este é positivo. Ciclo a depender linearmente do valor de n (número de linhas). Logo O(n) e Ω(1).
- Cálculo da maior coluna do retângulo (função **getcIndex(board b)**). Ciclo a depender do número de linhas n. Logo O(n).

- Cálculo do maior quadrado (função **getMaxSquareSide(board b)**). Ciclo a depender da subtração do índice da maior coluna (número constante) ao número de linhas n. Logo O(n).

- Retirar quadrado do retângulo (função **deleteSquare(board b, int max)**). Ciclo dependente do índice da maior coluna e da dimensão do quadrado passado como argumento **max**. Logo O(n).

- Criação de uma cópia de um **board** (função **createCopy(board b)**). Ciclo idêntico ao da função **createBoard()**. Logo também Θ(n).

- Definição do operador **==** para comparar dois vetores de inteiros. Ciclo depende da dimensão dos vetores a comparar. Logo O(n).

- Definição de uma função hash para vetores de inteiros. Ciclo depende da dimensão dos vetores a calcular. Logo O(n).

Relatório 1º projecto ASA 2022/2023

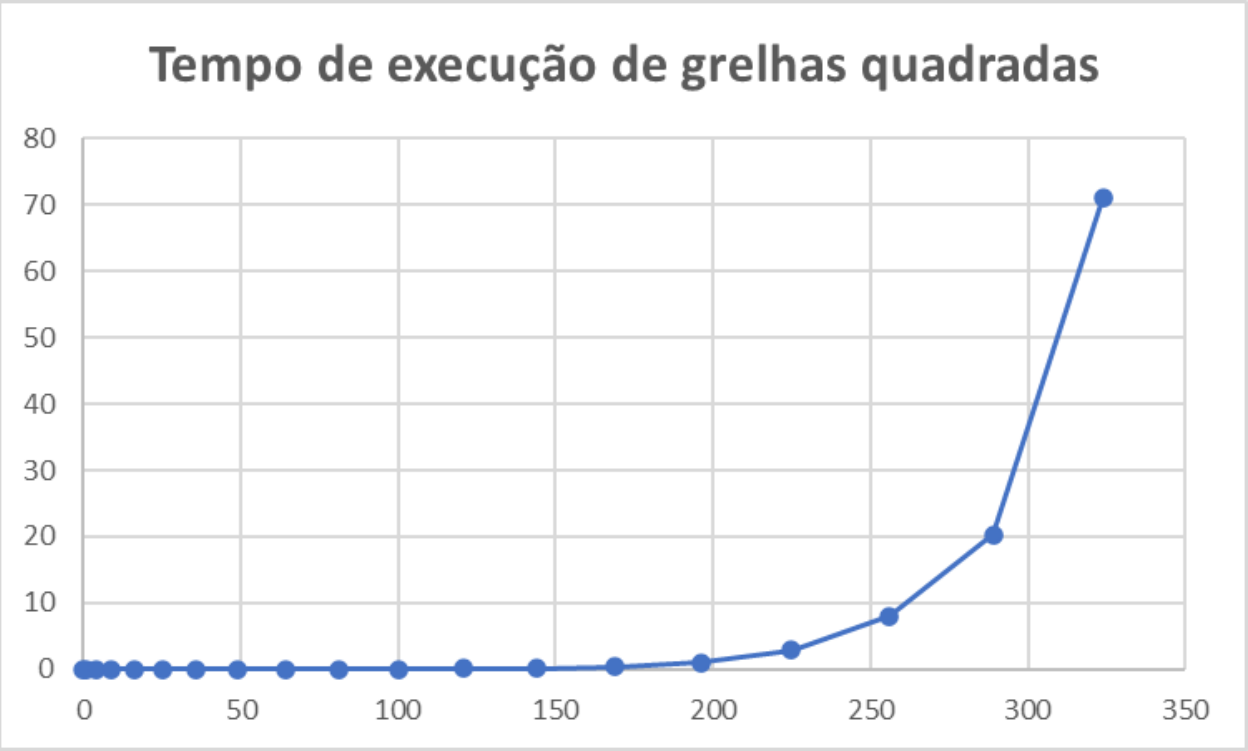
Grupo: TP049

Alunos: Bernardo Galante (102423) e Henrique Machado (103266)

Avaliação Experimental dos Resultados

O pior caso para o cálculo de configurações é quando o número de linhas é igual ao número de colunas, ou seja, quando a área é quadrada.

Testamos o nosso programa com quadrados de dimensões 1 por 1 até 18 por 18 de modo a obter um gráfico do tempo para cada caso.



- eixo dos XXs: número de quadrados na área a ladrilhar.
- eixo dos YYs: tempo de execução em segundos.

Claramente o gráfico gerado cresce exponencialmente com o número de quadrados. Assim a complexidade da solução final é O(e^(n^2)).