RELATÓRIO 1º PROJETO ASA 2023/2024

01 Descrição do Problema e da Solução

O problema abordado consiste na otimização de uma linha de corte de chapas de mármore, de forma a maximizar o preço ao cortar uma chapa em peças menores, cada uma com um determinado preço e dimensão.

Na solução proposta optámos por uma abordagem de programação dinâmica, onde o algoritmo percorre as células da matriz e calcula o valor máximo possível para cada subproblema, utilizando os resultados pré-calculados armazenados. Cada entrada na matriz representa o preço máximo obtido para uma determinada combinação de largura e altura da peça. O resultado, ou seja, o preço máximo da chapa original, é guardado na célula que corresponde à dimensão total chapa de mármore.

02 Análise Teórica

Função recursiva da solução proposta:

$$C(i,j) = \begin{cases} C(i,j), & se\ i = j = 1 \\ \max\big(C(k,j) + C(i-k,j),\ C(i,k) + C(i,j-k), C(i,j)\big) & se\ 1 \le k \le \frac{i}{2}\ ou\ 1 \le k \le \frac{j}{2} \end{cases}$$

Seja *n* o número de tipos de peças que podem ser produzidas, e *X* e *Y* as dimensões da chapa de mármore, temos:

- Inicialização da matriz de programação dinâmica: percorre todas as células uma vez, ou seja, $\Theta(XY)$.
- Leitura do input: primeiramente, ocorre a leitura do tamanho da chapa inicial e, de seguida, sucede-se o ciclo para a leitura do tipo de peça, que depende linearmente de *n*, pelo que a complexidade da leitura é $\Theta(n)$.
- Aplicação do algoritmo: percorre a matriz e itera sobre todas as possíveis divisões da peça, vertical e horizontalmente, não ultrapassando metade da largura ou comprimento da combinação específica de largura (*i*) e comprimento (*j*), escolhendo o corte de valor máximo. Assim, a complexidade é $O(XY \cdot \max(X,Y))$.
- Apresentação dos dados: como referido anteriormente, o resultado pode ser obtido simplesmente acedendo à última célula da matriz, $\Theta(1)$.

```
C is the Matrix
function calculatePrice(piece, C)

    for i from 1 to piece_width

        for j from 1 to piece_height

            maxPrice =  C[i][j]

            for k from 1 to max(i/2, j/2)
                if k <= i/2
                    maxPrice = max(maxPrice, C[i - k][j] + C[k][j])
                if k <= j/2
                    maxPrice = max(maxPrice, C[i][j - k] + C[i][k])

            C[i][j] = maxPrice

    return C[piece_width][piece_height]
```

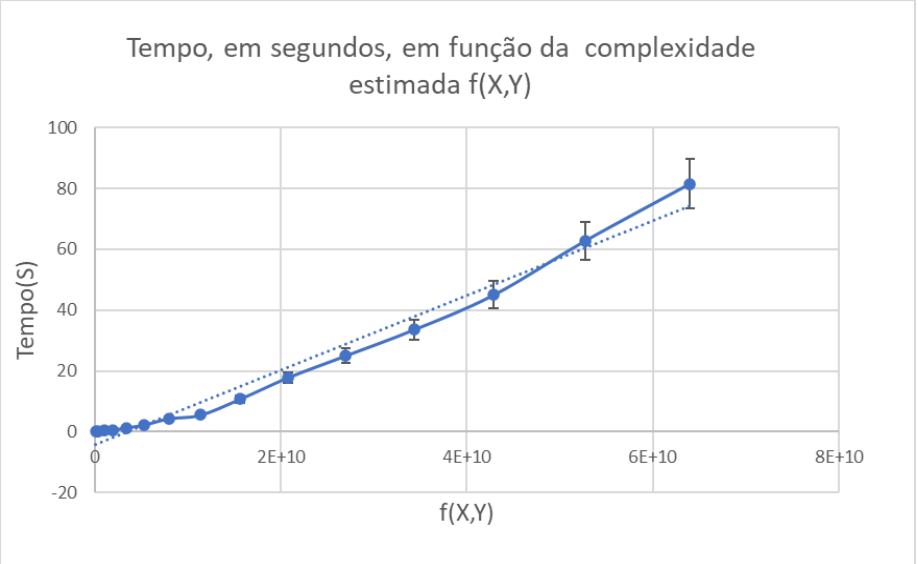
Assim a complexidade global da solução é $O(XY \cdot \max(X,Y))$ devido ao preenchimento da matriz de programação dinâmica. As restantes operações contribuem com uma complexidade constante em comparação com o preenchimento da matriz.

03 Avaliação Experimental dos Resultados

Para a avaliação experimental foram criadas 15 instâncias de tamanho incremental, sendo cada instância uma chapa quadrada**,** em que X e Y são incrementados 250 unidades a cada instância. Consequentemente $X+Y$ registam um aumento de 500 unidades comparativamente à instância anterior. Relativamente ao tipo de peças, o número testado manteve-se constante, utilizando, em todos os casos, 500 tipos.

A escolha de uma chapa quadrada deve-se a este ser o pior caso possível, pois quando $X = Y$ a complexidade é necessariamente O(X^3).

Abaixo estão os resultados obtidos:



Como é possível concluir, o tempo não varia linearmente com a dimensão da chapa. No entanto, este resultado está de acordo com a previsão teórica, já que existe uma relação linear entre o tempo e f(X,Y).