Relatório 1º projeto ASA 2023/2024

Grupo: AL002
Alunas: Cecília Correia (106827) e Luísa Fernandes (102460)

Descrição do Problema e da Solução

O **problema** consiste em otimizar o valor (preço) obtido ao cortar uma chapa. A chapa pode ser cortada vertical e horizontalmente. O objetivo é determinar os cortes de forma a maximizar o valor total das peças. A **solução** proposta utiliza programação dinâmica, constrói uma matriz (mtz) que representa a chapa dada no input. Esta matriz é preenchida considerando todos os cortes verticais e horizontais e os valores das diferentes peças disponíveis. O algoritmo calcula iterativamente os valores ótimos para cada posição na matriz. É uma abordagem eficiente, pois considera todos os cortes possíveis da chapa e escolhe a melhor combinação. No final, o valor máximo acumulado na posição (mtz[X][Y]) da matriz representa o resultado desejado.

Análise Teórica

Pseudo-código que ilustra a função recursiva da solução proposta:

```
MaximizeValue(X, Y, mtz)
    // Iteração sobre as dimensões da chapa
    for i = 1 to X
        for j = 1 to Y
            // Caso 1: Não cortar a chapa horizontalmente
            mtz[i][j] = max(mtz[i][j], mtz[i - 1][j]
            // Caso 2: Não cortar a chapa verticalmente
            mtz[i][j] = max(mtz[i][j], mtz[i][j - 1]
            // Caso 3: Cortar a chapa horizontalmente
            for k = 0 to i - 1
                mtz[i][j] = max(mtz[i][j], mtz[k][j] + mtz[i - k][j]);
            // Caso 4: Cortar a chapa verticalmente
            for k = 0 to j - 1
                mtz[i][j] = max(mtz[i][j], mtz[i][k] + mtz[i][j - k])
            endfor
            endfor
        endfor
    endfor
    return mtz[X][Y]
```

A leitura dos dados de entrada envolve apenas operações de leitura simples, sendo linear em relação ao número de peças, N, que vai ser recebido no input, mais os dois inputs iniciais obrigatórios (X e Y do tamanho da chapa e o número de peças N). Fica complexidade O(2 + N) que pode ser arredondada a O(N), pois para valores grandes de N, o 2 passa a ser desprezado. Logo: O(N).

A inserção dos dados na matriz é feita em dois loops que vão de 1 até X e Y.
A iteração externa (for i) ocorre X vezes. Logo: O(X)
A iteração interna (for j) ocorre Y vezes para cada iteração externa. Logo: O(Y)
Dentro do loop de j, os loops internos dos casos 3 e 4 têm complexidade O(i) e O(j), respetivamente, que no pior caso (na última iteração), correspondem a X - 1 e Y - 1. Logo: O(X + Y)
A complexidade global é dada pelo produto das complexidades de cada nível de iteração: O(X * Y * (X + Y)).

Avaliação Experimental dos Resultados

Gerámos 13 instâncias de tamanho incremental da chapa, com um número de peças constante (1), e a peça também constante (2 3 1). Corremos o código usando time para saber o tempo de execução, que registámos na tabela seguinte (coluna da direita):

				tempo real
x*y*(x+y)	x	y	x+y	s
54000000	300	300	600	6.572
128000000	400	400	800	6.964
182250000	450	450	900	8.7
250000000	500	500	1000	9
332750000	550	550	1100	10.437
432000000	600	600	1200	11.46
843750000	750	750	1500	16.524
1458000000	900	900	1800	22.601
2000000000	1000	1000	2,000	26.437
3456000000	1200	1200	2400	44.699
6750000000	1500	1500	3000	86.069
11664000000	1800	1800	3600	134.216
16000000000	2000	2000	4000	172.376

O gráfico gerado do tempo de execução em função de X + Y, deu uma função exponencial (Figura 1):

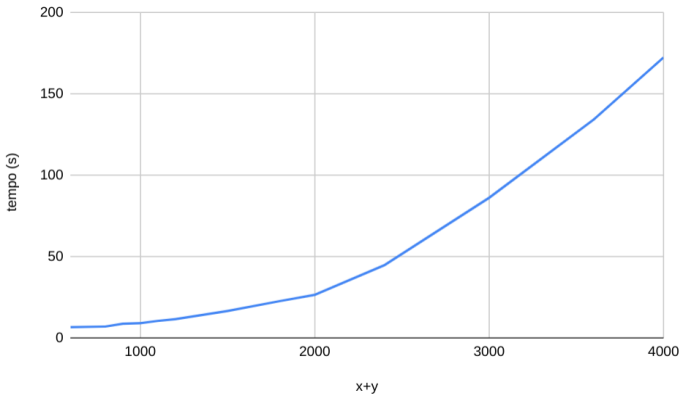

Figura 1

À medida que as dimensões da chapa aumentam, a quantidade de subproblemas a ser considerados cresce exponencialmente, o que está de acordo com a análise teórica prevista. Quando alteramos o gráfico para representar a função f(X, Y)= X * Y *(X + Y) no eixo dos XX, o que corresponde à complexidade do programa, prevista teoricamente, obtemos uma relação linear com os tempos de execução utilizados anteriormente (Figura 2). Isto confirma que a nossa implementação está de acordo com a análise teórica de O(X * Y * (X + Y)).

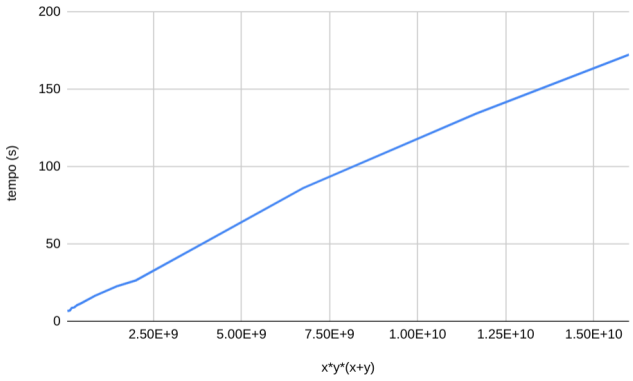

Figura 2