Relatório 1º projeto ASA 2023/2024

Grupo: TP049
Aluno(s): Henrique Machado (103202) e Vasco Pereira (103368)

Descrição do Problema e da Solução

Criámos uma função para dar parse ao input através do stdin. Nesta função começamos por criar um tuple *dimensoes* onde guardamos as dimensões da chapa. De seguida guardamos o número de peças na variável *n_cuts*. Criámos uma tabela chamada *plates_value* de tamanho *dimensoes.0* por *dimensoes.1* onde *plates_value[1][1]* é o valor máximo que conseguimos obter com os nossos cortes para uma chapa de tamanho 1 por 1. Esta tabela é inicializada a 0. De seguida lemos *n_cuts* linhas do stdin sendo cada linha uma peça possível, para cada linha verificamos se esse corte e o seu rodado cabem na chapa se couberem verificamos se já valem mais que o valor que está em *plates_value* para essas dimensões, se tiver um valor mais alto inserimos o novo valor em *plates_value[i][j]* sendo i uma dimensão da peça e j a outra.

De seguida criámos a função *maximize_plate_value* que recebe o comprimento da chapa, a sua largura e a tabela *plates_value*. Esta função itera i de 1 ao comprimento da chapa inclusive, para cada i itera j de 1 à largura da chapa inclusive. Estes dois loops farão com que possamos simular todos os tamanhos de chapa até ao tamanho original. Para cada tamanho de chapa teremos um loop que itera k de 0 a $i/2$ simulando todos os cortes verticais possíveis, para cada corte testamos se o valor das nossas duas novas chapas são maiores que o valor da chapa inteira. Para cada tamanho temos também um loop que itera l de 0 a $j/2$ desta vez simulando os cortes horizontais possíveis e fazendo a mesma verificação. No final retorna $plates\_value[dimensoes.0][dimensoes.1]$ que é o melhor valor para a chapa total.

Análise Teórica

Solução proposta

$$dp[i][j] = max(dp[i][j],\ dp[i-k][j] + dp[k][j],\ dp[i][j-l] + dp[i][l])$$

Onde *dp* é a tabela *plates_value*.

Pseudo Código com complexidade

- Simples leitura com alocação e inicialização da estrutura de dados a 0, com 1 ciclo a depender linearmente do número de peças. Logo, $\mathcal{O}(num\_pecas)\Big(\mathcal{O}(n)\Big)$.
- Aplicação do algoritmo indicado para o cálculo da função recursiva. Temos 3 nested loops logo $\mathcal{O}(n^3)$ admitindo que ambas as dimensões da placa são iguais e de nome n.
- Apresentação dos dados $\mathcal{O}(1)$

Complexidade global da solução: $\mathcal{O}(n^3)$

Relatório 1º projeto ASA 2023/2024

Grupo: TP049
Aluno(s): Henrique Machado (103202) e Vasco Pereira (103368)

Avaliação Experimental dos Resultados

Como analisada antes a complexidade temporal do algoritmo apenas depende da dimensão da chapa ou seja de X e Y, por isso foram geradas 10 instâncias de tamanho 500x500 incrementando em intervalos de 500, podemos observar o gráfico de tempo no Gráfico 1.

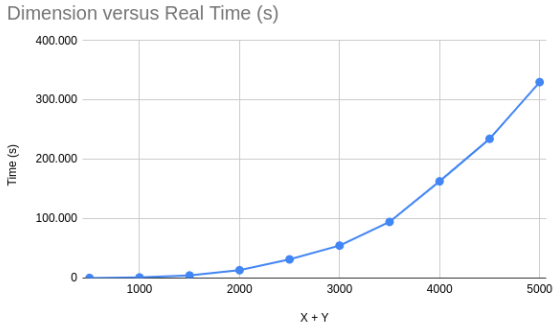

Figure 1: Gráfico 1

Como conseguimos ver o gráfico gerado está de acordo com a complexidade teórica sendo que o tempo de execução é cúbico em relação às dimensões da chapa. Logo se pusermos o eixo dos X a variar com a complexidade prevista na analise teórica ($\mathcal{O}(n^3)$) o gráfico será linear como pode ser visto no gráfico 2.

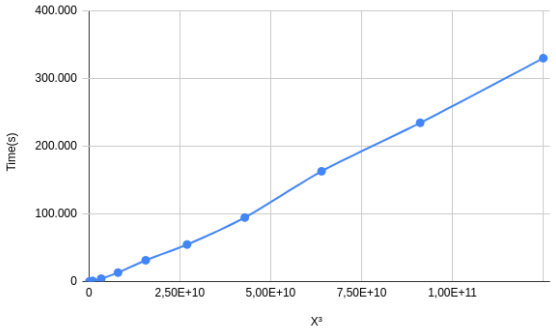

Figure 2: Gráfico 2

Assim conseguimos ver que temos uma relação linear com os tempos no eixo dos Y, confirmando que a nossa implementação está de acordo com a analise teórica.