Relatório 1º projeto ASA 2023/2024

Grupo: AL008
Alunos: Vasco Conceição (106481), Henrique Luz (99417)

❖ Descrição do Problema e da Solução

A nossa solução utiliza a técnica de programação dinâmica. Ao processar o input, começamos por criar uma tabela de x (comprimento da chapa) linhas e y (largura da chapa) colunas com todos os elementos a 0. Depois, por cada peça de dimensões $[i, j]$ lida, atribui-se o preço de venda da peça ao elemento da tabela na posição $[i, j]$. Passando agora para o algoritmo em si, a ideia é, quando acabar o algoritmo, termos os valores máximos a obter por uma chapa de dimensões $[i, j]$ precisamente na posição $[i, j]$ da tabela (a partir daqui vamos chamar a esta tabela de v).

Ao percorrer a tabela, v, de cima para baixo, da esquerda para a direita, comparamos o valor de $v[i, j]$, isto é, o preço da peça de dimensões $[i, j]$, que é 0 caso não seja pedida, com o máximo entre $v[i, k]$ + $v[i, j$ - $k]$ (que corresponde ao corte vertical da chapa) e $v[k, j]$ + $v[i$ - $k, j]$ (que corresponde ao corte horizontal da chapa). Desta forma, escolhemos para $v[i, j]$ o maior valor entre entregar a peça ao cliente ou cortá-la.

❖ Análise Teórica

Função recursiva da solução proposta:

$$v(i,j) = \begin{cases} 0 & \text{se } i = 0 \vee j = 0 \\ max_{1 \le k_1 \le i/2+1, 1 \le k_2 \le j/2+1} \{v[i,j], v[k_1, j] + v[i - k_1, j], v[i, k_2] + v[i, c - k_2]\} & c.c. \end{cases}$$

De notar que utilizámos os limites superiores i/2+1 e j/2+1, dado que o resultado de cortar a chapa $[i, j]$ considerando k = l em $[i, l]$ e $[i, j$ - $l]$ é igual a cortar a mesma chapa considerando $k = j - l$ em $[i, j - l]$ e $[i, j - (j - l)]$.

Pseudocódigo:

```
parseDimensions():
  read x, y from input
  v = create 2D vector of size (x + 1) by (y + 1) initialized with zeros
parsePrice():
  read n from input
  for k from 0 to n do
    read i, j, p from input
    if i <= x and j <= y then
      v[i][j] = p
      if j <= x and i <= y then
        v[j][ i] = p
computeMaxValue():
  for i from 1 to x do
    for j from 1 to y do
      for k from 1 to i/2 + 1 do
        v[i][j] = max(v[i][j], v[k][j] + v[i - k][j])
      for k from 1 to j/2 + 1 do
        v[i][j] = max(v[i][j], v[i][k] + v[i][j - k])
```

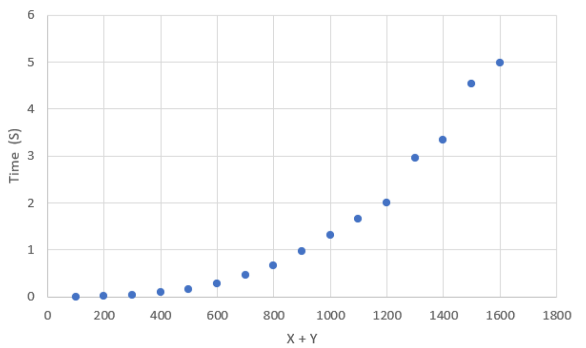
Grupo: AL008

Alunos: Vasco Conceição (106481), Henrique Luz (99417)

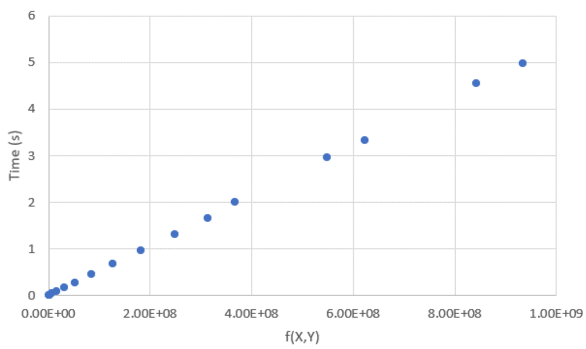
1. Leitura dos dados de entrada: Simples leitura do input, com um ciclo a depender linearmente de n (número de peças). Logo, O(n).
2. Processamento da instância: Inserir o preço da peça na tabela v. Logo, O(1).
3. Aplicação do algoritmo indicado para cálculo da função recursiva: Percorre-se a tabela v utilizando dois ciclos *for* apropriados, cujo custo é xy. Para cada entrada da tabela, são percorridas, em média, $x/4$ linhas e $y/4$ colunas, sendo o custo por entrada $x/4+y/4$ = O($x+y$). Logo, o custo total da aplicação do algoritmo é O($xy(x+y)$).
4. Apresentar o resultado final: é feito um print do valor que se encontra na última posição da tabela, $v[x][y]$. Corresponde a uma complexidade O(1).
5. Complexidade global da solução: O(n) + O(1) + O($xy(x$+y)) + O(1) = O($xy(x+y)$). Vale notar que pode haver um caso raro em que $n > xy(x+y)$, mas de forma geral, admitindo que isso não acontece, o custo está na aplicação do algoritmo e não no processamento do input.

❖ Avaliação Experimental dos Resultados

Neste gráfico, apresentamos o tempo de execução do algoritmo em função do tamanho ($x+y$) da entrada. Para tal, utilizámos 16 instâncias espaçadas 100 de tamanho entre si.



O tempo de execução não é linear nas dimensões da chapa. Assim, para determinar se a previsão pela análise teórica é correta, vamos pôr o eixo dos XX a variar com a quantidade prevista pela análise teórica.



Ao mudarmos o eixo dos XX para $f(x, y)$ = O($xy(x+y)$), vemos que temos uma relação linear com os tempos no eixo dos YY, confirmando que a nossa implementação está de acordo com a análise teórica de O($f(x, y)$).