Relatório do 1º Projeto ASA 2024/2025

Grupo: AL019
Alunos: João Matreno (110246) e Samuel Gomes (110274)

Descrição do Problema e da Solução

Para encontar a parentisação mais á esquerda de forma para a qual a expressão tem o valor desejado vamos começar por resolver os sub-problemas primeiro em ordem crescente de forma a chegarmos ao problema final. Vamos armazenar os sub-problemas numa tabela $m \times m$ em que cada entrada B_{ij} contem os valores possíveis da expressão, E, do elemento i ao j. B_{ij} será um vetor de elementos do tipo (v, l, k, r), em que v é o valor que a expressão pode ter de i a j, resultado da operação $(B_{ik}[l] \oplus B_{k+1j}[r])$.

B_{11}	B_{12}	B_{13}	$\cdots$	B_{1m}
	B_{22}	B_{23}	$\cdots$	B_{2m}
		B_{33}	$\cdots$	B_{3m}
			$\ddots$	$\vdots$
				B_{mm}

Por exemplo:

2 4
1 2
2 1
1 2 2 1
1

1			
	2		
		2	
			1

1	2		
	2	1	
		2	2
			1

1	2	1	
	2	1	2
		2	2
			1

$B_{13} = ((B_{12} \oplus B_{33}) \vee (B_{11} \oplus B_{23})) = 1$, guardamos só a primeira, porque é a parentisação mais á esquerda. O mesmo se aplica para B_{24}. $B_{14} = ((B_{13} \oplus B_{44}) \vee (B_{12} \oplus B_{34}) \vee (B_{11} \oplus B_{24})) = 1 \vee 2$, como a opção do meio nos dá um resultado que já obtivemos anteriormente não a guardamos. Para obtermos o resultado final fariamos uma recurssão do tipo: $1 = B_{14} = (B_{13} \oplus B_{44}) = ((B_{12} \oplus B_{33}) \oplus 1) = (((1 \oplus 2) \oplus 2) \oplus 1)$

Análise Teórica

Algorithm 1: Algoritmo de Construção da Tabela

for $i \in \{1, \ldots, m\}$ **do**
  $B_{ii} \leftarrow B_{ii} \cup E_i$;
for $d \in \{1, \ldots, m-1\}$ **do**
  for $i \in \{1, \ldots, m-d\}$ **do**
    $j \leftarrow i + d$;
    $adds \leftarrow 0$;
    $\vec{A}$ is a vector of size n all initiated at $false$;
    for $k \in \{j-1, \ldots, i\}$ *(decreasing), if* $adds \neq n$ **do**
      for $l \in \{1, \ldots, B_{ik}.size\}$, *if* $adds \neq n$ **do**
        for $r \in \{1, \ldots, B_{k+1j}.size\}$, *if* $adds \neq n$ **do**
          $v \leftarrow (B_{ik}[l] \oplus B_{k+1j}[r])$;
          if $\vec{A}_v == false$ **then**
            $B_{ij} \leftarrow B_{ij} \cup (v, l, k, r)$;
            $\vec{A}_v \leftarrow true$;
            $adds \leftarrow adds + 1$;

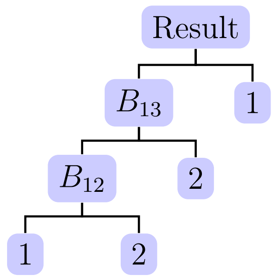
Relatório do 1º Projeto ASA 2024/2025

Grupo: AL019
Alunos: João Matreno (110246) e Samuel Gomes (110274)

Leitura dos dados de entrada: simples leitura do input, com ciclos a depender quadraticamente de n (tabela de operaçoes, M) e linearmente m (expressãao, E). Logo, $O(n^2 + m)$

O preenchimento da tabela depende da quantidades de células a preencher, $m(m+1)/2$, e do custo de preencher cada tabela, que será o número máximo de elementos de cada entrada, n. Logo $O(nm^2)$.

A recurssão que constroí a resposta final é chamada $m+m-1$ vezes, pois cada dois números requerem dois parentesis e geram um novo número, como se pode ver no esquema do exemplo utilizado anteriormente. Cada chamada tem um custo constante. Logo $O(m)$.



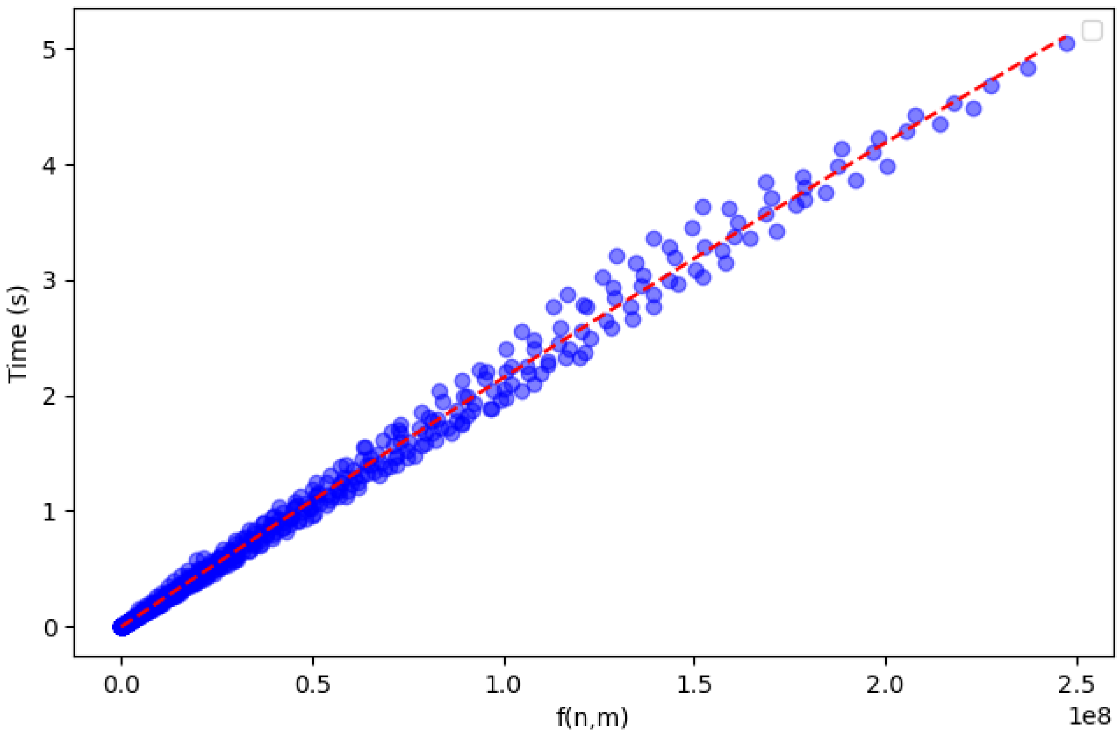
Apresentação dos dados depende linearmente do tamanho da expressão. Logo, $O(m)$.

Complexidade global da solução: $O(nm^2)$.

Avaliação Experimental dos Resultados

Para fazer a análise experimental do nosso código medimos o tempo que demorava para encontrar a solução para inputs de diferentes tamanhos, tal que:

$$n = 1 + 5k_1,\ k_1 \in \mathbb{Z},\ 0 \leq k_1 \leq 25$$
$$m = 1 + 70k_2,\ k_2 \in \mathbb{Z},\ 0 \leq k_2 \leq 22$$



Neste gráfico temos que $\mathbf{f(n,m) = nm^2}$ e como podemos ver o tempo cresce linearmente em função de $f(n,m)$, o que confirma o esperado pela análise teórica de que a complexidade temporal do algoritmo é tal que $\mathbf{T(f(n,m)) \in O(nm^2)}$.