Grupo: AL092 | Alunos: Mariana Carvalho; 109974; Beatriz Teixeira, 110263

Relatório do 1º projeto ASA 2024/2025

DESCRIÇÃO DO PROBLEMA E DA SOLUÇÃO

A solução desenvolvida para auxiliar o arqueólogo João Caracol na decifração das operações encontradas no Templo Perdido utiliza um algoritmo eficiente e dinâmico. O problema consiste em identificar a correta parentização das sequências de operações com o operador $\oplus$, com base na matriz de resultados fornecida no input, a partir da segunda linha. O objetivo é garantir que as combinações de agrupamentos reproduzam o resultado esperado, indicado no último campo do input.

O algoritmo começa por validar os cálculos intermédios utilizando a matriz de resultados. Em seguida, analisa todas as possíveis combinações de agrupamento das operações, verificando os resultados de cada uma. Sempre que possível, dá prioridade à parentização mais à esquerda que satisfaça o valor final esperado, garantindo precisão e eficiência no processo.

ANÁLISE TEÓRICA

Fórmula recursiva:

$$T[i][j] = \begin{cases} \{\{0,1, operations[i], se\ i = j \\ \bigcup_{k=i}^{j-1} \{combine(T[i][k], T[k+1][j])\}, caso\ contrário \end{cases}$$

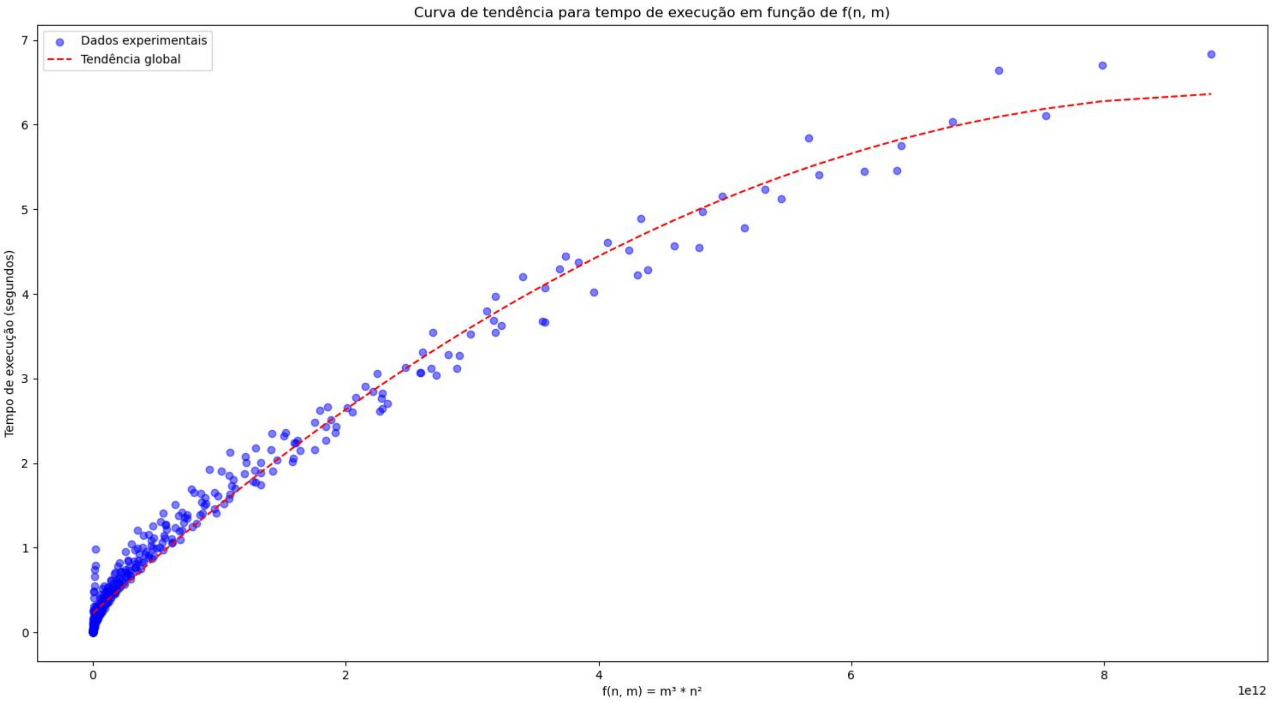
```
for i := 1 to m - 1
    resultTbl->[i,i] := [{0, o[i]}]
for d := 1 to m - 1
    for i := 1 to m - d
        let j := i + d
        clear currentResult
        clear set{uniqueResults}
        set foundAllResults := false
        for k := j - 1 to i and not foundAllResults
            let leftSet := resultTbl[i,k]
            let rightSet := resultTbl[k+1,j]
            for each left in leftSet
                let leftResult := last value of left
                for each right in rightSet
                    let rightResult := last value of right
                    compute combinedResult := tbl[leftResult-1,
rightResult-1]
                    if d == m - 1 and combinedResult == result
                        set finalResult := [{currentIndex, left, right, k,
combinedResult}] to currentResult
                        exit all loops
                    if combinedResult not in set{uniqueResults}
                        add combinedResult to set{uniqueResults}
                        append {currentIndex, left, right, k,
combinedResult} to currentResult
                    if size of set{uniqueResults} == n
                        set foundAllResults := true
                        exit loop
        set resultTbl[i,j] := currentResult
set resultTbl->[1, m] := finalResult
```

- Leitura do input: Primeiramente, o programa lê os valores n (referente ao tamanho da matriz de resultados) e m (tamanho da sequência de operações), seguidos pela matriz de resultados(com n linhas), a sequência de inteiros e o resultado esperado. Complexidade: **O(n^2 + m)**, considerando a leitura da tabela(n*n) e da sequência(m).
- Construção da tabela de combinações: A tabela armazena os resultados intermediários para cada subcadeia (com tamanho igual a d) da sequência. Complexidade: **O(m^3*n^2)**, já que cada par de índices (i, j) da sequência pode gerar uma subcombinação.
- Reconstrução recursiva dos parênteses: Utiliza a tabela de combinações para reconstruir, de forma recursiva, a parentização dando prioridade à solução com os parênteses mais à esquerda. Complexidade: Cada chamada realiza uma concatenação de strings de tamanho L, o que adiciona O(L) por chamada. Como há O(m) chamadas, a complexidade total é O(m^2), assim a complexidade total da reconstrução é **O(m^2)**.
- Apresentação do resultado: Se a solução for encontrada, imprime 1, seguido pela solução com a parentização mais à esquerda. Caso contrário, imprime 0. Complexidade: **O(1)**.

Assim, a complexidade global da solução é **$O(m^3*n^2)$**, pois sobrepõem-se às outras complexidades. As restantes operações, como a reconstrução recursiva da parentização e a análise do input, contribuem com uma complexidade que cresce de forma mais lenta em comparação com o preenchimento da matriz dinâmica.

AVALIAÇÃO EXPERIMENTAL DOS RESULTADOS

Para realizar a avaliação experimental, foram efetuados 722 testes, variando n de 5 a 100 com incrementos de 5, e para cada valor de n, o número de m's variou de 50 a 1000, com incrementos de 25.



O gráfico apresenta um comportamento logarítmico, e não linear, devido à estratégia de interrupção dos testes assim que n resultados são encontrados. Essa otimização reduz significativamente a complexidade dos loops *left* e *right*, que, em vez de atingirem $O(n^2)$ na maioria dos casos, aproximam-se de $O(n)$. Como resultado, o comportamento observado segue uma curva logarítmica.