Relatório 1º projecto ASA 2024/2025

Grupo: AL010
Alunos: Francisco Silva (110409) e Marta Braga (110034)

Descrição do Problema e da Solução

A solução desenvolvida usa programação dinâmica para preencher uma matriz $m \times m$ que denominámos dpTable. Inicialmente, a diagonal principal é preenchida com os valores da sequência dada. Em seguida, as restantes células da parte superior da dpTable são preenchidas diagonal a diagonal. Para cada célula determinamos os pares de subproblemas que a compõem e calculamos o resultado de cada par usando a tabela de operações binárias fornecida. Por fim, na célula simétrica, armazenamos a posição onde o parêntese é fechado, juntamente com os resultados dos subproblemas associados, para que possamos, posteriormente, reconstruir este resultado. É de notar, que, caso duas combinações de subproblemas deem o mesmo resultado, apenas o primeiro é armazenado, uma vez que apenas queremos a combinação que gerar parênteses mais à esquerda.

Preenchimento da Tabela de Programação Dinâmica:

```
for diagonal = 1 to m − 1 do
    for lin = 0 to m − diagonal − 1 do
        col := lin + diagonal
        Initialize buffer to false
        solutionCounter := 0
        for i = diagonal − 1 down to 0 do
            for each leftResult in dpTable[lin][lin + i] do
                if solutionCounter = n then
                    break
                end if
                for each rightResult in dpTable[lin + i + 1][col] do
                    if solutionCounter = n then
                        break
                    end if
                    if buffer[subResult − 1] then
                        continue
                    end if
                    subResult := operationTable[leftResult − 1][rightResult − 1]
                    Add subResult to dpTable[lin][col]
                    closeBracket := lin − 1
                    Add closeBracket, leftResult, rightResult to dpTable[col][lin]
                    Mark buffer[subResult − 1] as true
                    solutionCounter := solutionCounter + 1
                end for
            end for
        end for
    end for
end for
```

Backtrack / Reconstrução da Solução

```
procedure BACKTRACK(dpTable, lin, col, subResult, expression)
    if lin = col then
        expression := expression + to_string(dpTable[lin][col][0])
        return
    end if
    parenthesis, leftResult, rightResult := 0
    for i = 0 to dpTable[lin][col].size() − 1 do
        if dpTable[lin][col][i] = subResult then
            parenthesis := dpTable[col][lin][3 · i]
            leftResult := dpTable[col][lin][3 · i + 1]
            rightResult := dpTable[col][lin][3 · i + 2]
            expression := expression + "("
            BACKTRACK(dpTable, lin, parenthesis, leftResult, expression)
            expression := expression+ " "
            BACKTRACK(dpTable, parenthesis + 1, col, rightResult, expression)
            expression := expression + ")"
            break
        end if
    end for
end procedure
```

Uma vez tendo a célula superior direita da dpTable calculada, verificamos se a resposta desejada está contida nessa célula. Caso positivo, iniciamos o backtracking. Caso a célula esteja na diagonal principal, adicionamos o número à expressão final. Caso contrário, usamos os dados da célula simétrica para identificar os subproblemas antecedentes, adicionando os parênteses que delimitam as combinações até reconstruir toda a expressão recursivamente.

Análise Teórica

Leitura de input: $O(\max(n^2, m))$. Criamos uma tabela $n \times n$ e inicializamos a diagonal da dpTable com a sequência de tamanho m.
Preenchimento da tabela: $O(m^3 \times n^2)$. Percorrer a tabela tem custo m^2. O custo de preencher cada célula é $m \times n^2$, uma vez que, no máximo, existem $(m-1)$ pares de subproblemas e cada subproblema tem no máximo n soluções.
Reconstrução da solução: $O(m \times n)$. São feitas $2m$ chamadas recursivas e cada chamada custa n.
Complexidade temporal final: É de $O(m^3 \times n^2)$ uma vez que é a maior complexidade encontrada no código.
Complexidade espacial final: $O(m^2 \times n)$ que corresponde ao tamanho da dpTable.

Relatório 1º projecto ASA 2024/2025

Grupo: AL010
Alunos: Francisco Silva (110409) e Marta Braga (110034)

Avaliação Experimental dos Resultados

Os gráficos abaixo ilustram a relação entre o tempo de execução do código em segundos e a função $f(n, m) = m^3 \times n^2$, que corresponde à complexidade calculada. Para gerar os gráficos foram utilizados valores de n entre 2 e 22, a variar de 4 em 4 e valores de m entre 50 e 500, a variar de 25 em 25.

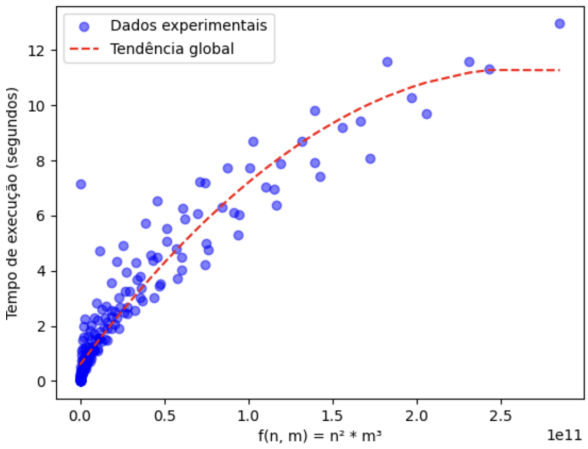

Gráfico 1: Teste aleatório

O gráfico 1 mostra ter uma tendência logarítmica. No entanto, isto deve-se a uma condição de paragem que foi implementada na dpTable para que, no preenchimento da tabela, os 3 loops mais interiores parem de executar quando as n soluções possíveis já tiverem sido encontradas.

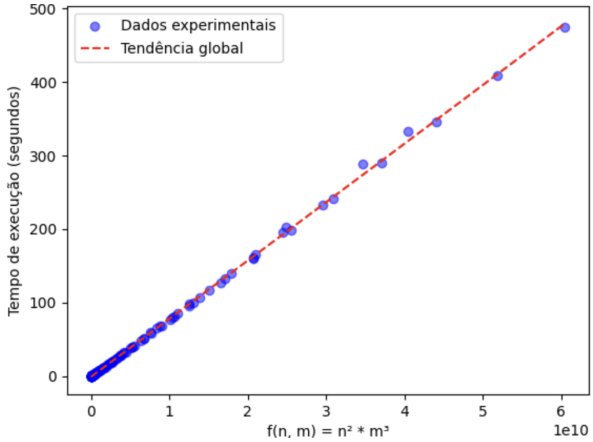

Gráfico 2: Teste sem heurístico

Contudo, no gráfico 2, já podemos observar dados experimentais que variam linearmente com a função calculada, uma vez que, nos testes utilizados para gerar este gráfico, a condição descrita acima nunca se aplica. Fazendo com que todas as combinações de subproblemas possíveis tenham obrigatóriamente de ser calculadas. Assim sendo, podemos confirmar que a implementação está de acordo com a análise teórica feita.