# Relatório 1º Projeto ASA 2025/2026

**Grupo:** AL030
**Alunos:** Leonor Costa Guedes (113396) e Manuel Francisco Santos Ramos Soares (113402)

---

## Descrição do Problema e da Solução

O problema consiste em determinar a ordem ótima para remover aminoácidos de uma cadeia, maximizando a energia libertada com base nos seus potenciais e afinidades bioquímicas. Trata-se de uma variante do problema clássico de multiplicação de cadeia de matrizes, adaptado para incluir uma tabela de afinidade entre classes de elementos.

A solução utiliza programação dinâmica para calcular a energia máxima em cada subintervalo, seguida de reconstrução com memorização para obter a sequência lexicograficamente menor. Esta abordagem mantém a eficiência do algoritmo clássico enquanto incorpora as especificidades do problema bioquímico.

---

## Análise Teórica

$$\mathrm{dp}[i,j] = \begin{cases} 0, & \text{se } i+1 \geq j \\ \max\limits_{\substack{k=i+1 \\ j-1}} (\mathrm{dp}[i,k] + \mathrm{dp}[k,j] + \mathrm{contrib}(i,k,j)), & \text{caso contrário} \end{cases}$$

onde,

$$\mathrm{contrib}(i,k,j) = \mathrm{P}[i] * \mathrm{A}[i][k] * \mathrm{P}[k] + \mathrm{P}[k] * \mathrm{A}[k][j] * \mathrm{P}[j]$$

● **Leitura dos dados de entrada:** leitura de n, dos valores P e de C, com ciclos que dependem linearmente de n
**Complexidade: O(n)**

● **Pré-processamento:** construção da matriz de afinidade A com tamanho $(n+2) * (n+2)$, envolvendo dois ciclos aninhados.
**Complexidade: O(n²)**

● **Aplicação de programação dinâmica:** preenchimento da tabela *dp* com três ciclos aninhados que dependem de n.
**Complexidade: O(n³)**

```
ler n; ler P[1..n]; ler C[1..n]
N := n + 2; inicializar P[0],P[N-1],C[0],C[N-1]
construir classIdx e AfTable
for i=0..N-1:
  for j=0..N-1:
    A[i][j] := afinidade entre i e j   // O(N^2)

inicializar dp[i][i] = dp[i][i+1] = 0
for len = 2..N-1:
  for i = 0..N-1en-1:
    j := i + len
    best := 0; found := false
    for k = i+1..j-1:
      contrib := P[i]*A[i][k]*P[k] + P[k]*A[k][j]*P[j]
      cand := dp[i][k] + dp[k][j] + contrib
      if not found or cand > best: best := cand; found := true
    dp[i][j] := best

memo := map vazio
function reconstruct(i,j):
  if i+1 >= j: return []
  if (i,j) in memo: return memo[(i,j)]
  target := dp[i][j]
  bestSeq := vazio
  for k = i+1..j-1:
    if dp[i][k] + dp[k][j] + contrib(i,k,j) != target: continue
    L := reconstruct(i,k)
    R := reconstruct(k,j)
    cand := L + R + [k]
    if bestSeq vazio or cand < bestSeq (lexico): bestSeq := cand
  memo[(i,j)] := bestSeq
  return bestSeq
```

● **Reconstrução da solução:** processo recursivo com memorização que, no pior caso, visita cada par (i, j) e realiza operações de concatenação.
**Complexidade: O(n³)**

● **Apresentação dos dados:** impressão do resultado e da sequência.
**Complexidade: O(n)**

# Relatório 1º Projeto ASA 2025/2026

**Grupo:** AL030
**Alunos:** Leonor Costa Guedes (113396) e Manuel Francisco Santos Ramos Soares (113402)

---

**Complexidade global da solução: O(n³)** - O termo cúbico domina sobre as restantes complexidades. Outras operações contribuem com complexidades assintoticamente inferiores, tornando-se insignificantes face ao custo da programação dinâmica.

---

**Avaliação Experimental dos Resultados**

O gráfico mostra a relação entre o tempo de execução do código e a função teórica $f(n) = n^3$, representada no eixo horizontal em milhões. Os valores de $n$foram escolhidos de forma que $n^3$varie de 0 até cerca de 8 milhões, permitindo comparar diretamente o crescimento experimental com o previsto pela análise teórica.

Observa-se uma tendência linear clara entre o tempo medido e $n^3$, indicando que o algoritmo apresenta, na prática, um comportamento cúbico. Isso confirma que todos os subproblemas foram computados sem interrupções antecipadas e que os resultados experimentais estão alinhados com a complexidade estimada.

| Tamanho (n) | Tempo (s) |
|---|---|
| 100 | 0.35 |
| 200 | 0.00 |
| 300 | 0.00 |
| 400 | 0.01 |
| 500 | 0.03 |
| 600 | 0.05 |
| 700 | 0.08 |
| 800 | 0.13 |
| 900 | 0.20 |
| 1000 | 0.26 |
| 1100 | 0.32 |
| 1200 | 0.41 |
| 1300 | 0.55 |
| 1400 | 0.75 |
| 1500 | 0.87 |
| 1600 | 1.17 |
| 1700 | 1.28 |
| 1800 | 2.02 |
| 1900 | 2.52 |
| 2000 | 3.09 |