Relatório 1º projeto ASA 2025/2026

Grupo: AL069
Alunos: Afonso Sítima(114018) e Rafael Dias(115156)

Descrição do Problema e da Solução

O objetivo é otimizar a quantidade de energia gerada ao retirar aminoácidos de uma sequência destes mesmos. Cada aminoácido é classificado em quatro categorias, e a energia libertada ao removê-lo varia conforme a sua própria categoria e a dos aminoácidos adjacentes, influenciada pela relação entre eles. Além disso, nas extremidades da sequência, há aminoácidos distintos com uma relação constante, que não depende dos demais.

A solução que adotamos consiste em usar programação dinâmica. Criamos duas matrizes:

1. Uma que armazena a maior energia possível para cada uma das sequência,

2. Outra que indica qual foi o último aminoácido retirado nessa parte.

Resolvemos o problema decidindo, para cada parte da sequência, qual aminoácido será o último a ser retirado, juntando as melhores soluções das partes à esquerda e à direita. A estrutura da matriz é tal que os problemas mais simples (sequências com um único aminoácido) estão na diagonal principal, e cada diagonal acima representa problemas gradualmente maiores. A linha indica o começo da sequência, e a coluna, o seu término; a linha e coluna zero ficam vazias, pois correspondem aos limites artificiais da sequência.

Análise Teórica

- **Leitura dos dados:** O(n), leitura de entrada simples com ciclos que dependem linearmente de n (número de aminoácidos na cadeia), esta leitura é feita duas vezes, uma para ler a energia dos aminoácidos e outra para a classe correspondente.

- **Processamento de Instâncias:** $O(n^2)$, criação das duas matrizes que armazenam os dados.

- **Programação Dinâmica:** $O(n^3)$, Utilização de três loops aninhados todos linearmente dependentes de n. Existem $O(n^2)$ subproblemas [i,j] e para cada um são testados até O(n) valores de k, logo a complexidade desta fase é $O(n^3)$,

```
para len de 1 até n:
    para i de 1 até n - len + 1:
        j ← i + len - 1
        best_val ← 0
        best_k ← 0
        para k de i até j:
            sub1 ← (k > i) ? best[i][k-1] : 0
            sub2 ← (k < j) ? best[k+1][j] : 0
            curr ← sub1 + sub2 + removal_energy(chain[i-1], chain[k], chain[j+1])
            se curr > best_val ou (curr = best_val e k > best_k):
                best_val ← curr
                best_k ← k
        fim para
        best[i][j] ← best_val
        choice[i][j] ← best_k
    fim para
fim para
```

Relatório 1º projeto ASA 2025/2026

Grupo: AL069
Alunos: Afonso Sítima(114018) e Rafael Dias(115156)

- Procura da Solução: O(n), reconstrução recursiva da sequência equivalente à maior energia, durante a recursão não é repetido nenhum subproblema logo a complexidade é O(n).

```
função build_order(i, j):
    se i > j:
        retornar
    k ← choice[i][j]
    build_order(i, k-1)
    build_order(k+1, j)
    adicionar k ao fim de order
```

- Funções extra: affinity O(1), removal_energy O(1).

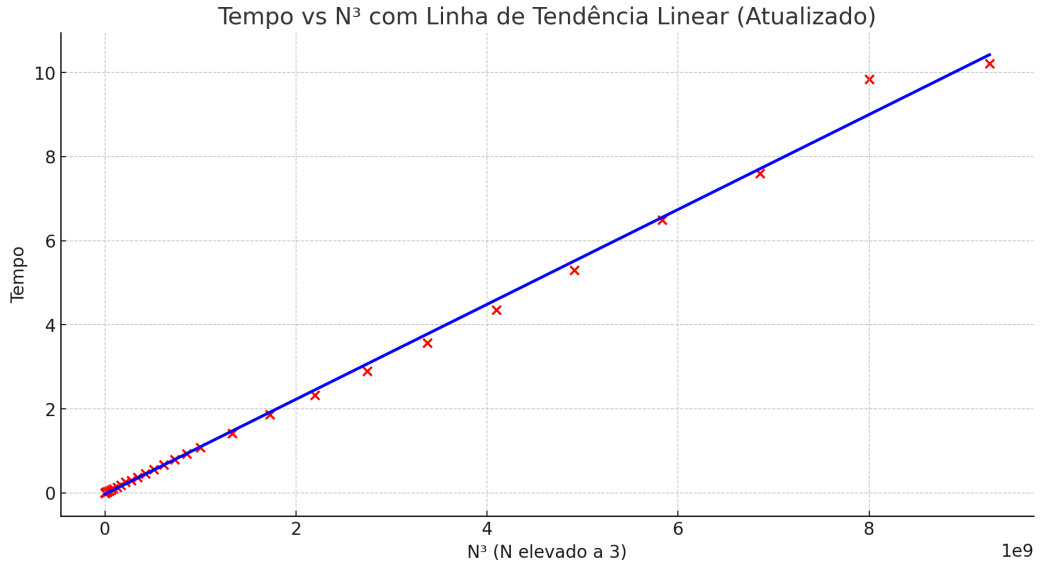
- Apresentação do resultado: O(n), equivalente ao número de dados de entrada.

A complexidade global prevista é $O(n^3)$, que é a complexidade da aplicação do algoritmo que descobre a solução pois as dos outros algoritmos no código crescem mais lentamente com o tamanho do input, logo a do algoritmo que preenche a matriz sobrepõe-se às outras.

Avaliação Experimental dos Resultados

Para realizar a avaliação experimental, foram efetuados 39 testes com n a variar entre 50 e 2100 com incrementos variados.

Eixo X: n^3 (complexidade do programa) | Eixo Y: Tempo de execução (em segundos).



Como podemos observar, o gráfico de tempo em função de n^3 apresenta uma tendência linear, confirmando a complexidade $O(n^3)$ que previmos.