**Descrição do Problema e da Solução**

O 1º projeto de ASA 2021/2022 é composto por dois problemas. Segundo o enunciado, o problema 1 consiste, dada uma sequência de inteiros, em determinar o tamanho da sua maior subsequência estritamente crescente (LIS), bem como o número de subsequências estritamente crescentes de tamanho máximo. Já o problema 2 resume-se, dadas duas sequências de inteiros, apenas a calcular o tamanho da maior subsequência comum estritamente crescente entre as duas (LCIS). Ambas as nossas soluções, implementadas em linguagem C++, recorrem a uma estrutura personalizada, composta por um vetor de inteiros e um inteiro que representa o tamanho desse vetor, para a representação das sequências e resolução dos problemas.

A solução do problema 1 é facilmente determinada com recurso a programação dinâmica e a tabulação, com dois vetores/tabelas auxiliares que guardam os resultados dos seus subproblemas (referentes ao tamanho e ao número de LIS até, e inclusive, cada índice da sequência inicial). Dadas duas variáveis (atualizadas aquando a descoberta de uma LIS com um novo tamanho máximo, ou, de uma LIS de tamanho igual à então descoberta) que representam a solução do problema, a ideia é iterar sobre todos os elementos da sequência inicial (ciclo exterior) e, para cada índice, percorrer todas as posições que estão para trás do mesmo (ciclo interior), podendo ser tomada ação sempre que um inteiro anterior for menor do que o em questão (uma nova IS é possível). Após essa verificação, comparando ao resultado do subproblema do índice do ciclo interior incrementado a 1, se o tamanho atual da LIS do índice do ciclo exterior for: menor, o tamanho é atualizado (num dos vetores/tabelas auxiliares); mas se for igual, ao número de LIS é adicionado o número de LIS do índice do ciclo interior (no outro vetor/tabela auxiliar).

O problema 2 (...)

**Análise Teórica**

Nesta análise teórica, considera-se N como o tamanho da primeira sequência de inteiros, e M como o tamanho da segunda (problema 2).

* Leitura dos dados de entrada e construção das sequências: simples leitura do input, com um ciclo a depender linearmente de N (problema 1)/ N + M (problema 2). Logo Θ(N)/ Θ(N + M).
* Resolução do problema 1: criação e inicialização dos vetores/tabelas auxiliares a 1 [O(N)]; resolução dos seus subproblemas e preenchimento dos vetores/tabelas [ciclo exterior que percorre a sequência inicial e ciclo interior que itera sobre o índice do ciclo exterior até ao início da sequência inicial O(N\*N), verificações e atualizações das soluções dos subproblemas e da solução do problema O(1)]. Logo, O(N\*N)
* Resolução do problema 2: . Logo, O(??)
* Apresentação dos dados. O(1)

Complexidade global da solução: O(N\*N) (problema 1)/ O(!??!) (problema 1)

**Avaliação Experimental dos Resultados**

As experiências do problema 1 foram realizadas num terminal bash do sistema operativo macOS, enquanto que as experiências do problema 2 (...).

Chart, line chart

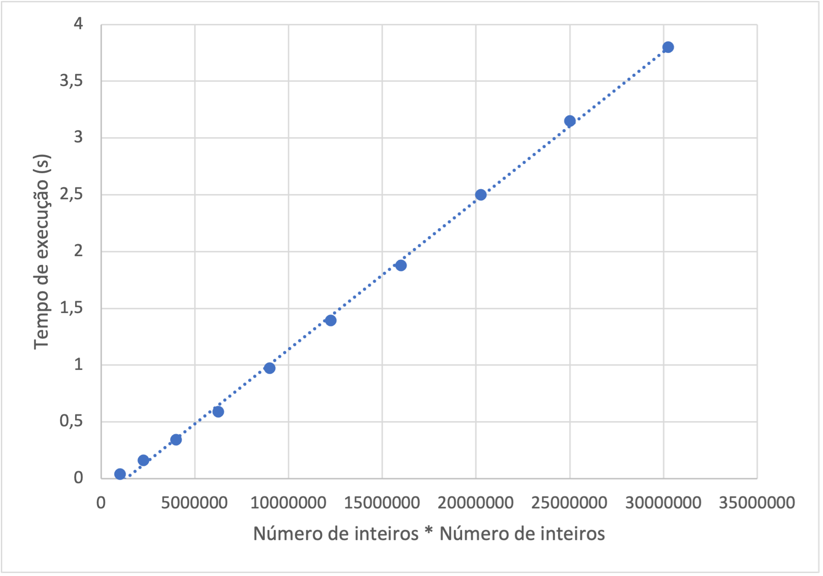
Description automatically generatedPara cada experiência foram gerados 10 grafos de tamanho incremental. De seguida, foi cronometrado o tempo de execução do programa para cada um dos grafos gerados. Como resultado, foram originados os gráficos da Figura 1 e da Figura 2.

Figura 2 - Gráfico de Tempo de Execução em função de N\*N

Figura 1 - Gráfico de Tempo de Execução em função de N\*N

Sendo que as linhas de tendência linear dos gráficos (com o tempo de execução em função da complexidade do respetivo problema) se revelam bastante próximas de todos os pontos, nos dois casos, pode-se concluir que os gráficos gerados para os diferentes problemas estão concordantes com análise teórica acima descrita, pois é possível observar que o tempo de execução do programa cresce linearmente com a complexidade apontada. Logo, a complexidade global da solução de cada problema verifica-se: O(V\*V) (problema 1)/(...).