Relatório 1º projecto ASA 2021/2022

Grupo: al054
Aluno(s): Diogo Falcão (99199) e Yassir Yassin (100611)

Descrição do Problema e da Solução

A implementação do programa foi elaborada em linguagem C++.

Problema 1- Dada uma sequência de inteiros, o algoritmo pretende calcular tanto o tamanho da sua maior subsequência estritamente crescente (LIS), bem como o número de LIS's desse tamanho. Para este fim foi utilizado uma solução que tem como objetivo ir armazenando o tamanho de todas as LIS's num array de inteiros, seguidamente é retornado o valor máximo deste array e quantas vezes este valor se repete.

Problema 2- Dadas duas sequências de inteiros, o algoritmo pretende calcular o tamanho da maior subsequência comum estritamente crescente (LCIS). Para solucionar este problema foi utilizada uma solução que tem como objetivo ir armazenando num array os valores do tamanho da LCIS para cada tamanho do segundo vetor dado como input, seguidamente é retornado o maior valor do array.

Análise Teórica

Problema 1- É inicializado um vetor de inteiros, X, onde são guardados os valores do input.

Para resolver o problema, começamos a ler os dados de input com ciclos lineares em Θ(N), de seguida é guardado o tamanho do vetor X num inteiro, N. São ainda criados 2 vetores, comps e numbersubs, ambos de tamanho N sendo a numbersubs inicializada com as suas entradas a 1, O(N). Aplicando o algoritmo, preenche-se o vetor comps e numbersubs percorrendo os elementos de X, e para cada elemento de X, percorre-se novamente o vetor X começando no índice à frente, apanhando todos os valores maiores e guardando os tamanhos máximos no vetor comps e o número de subsequências no vetor numbersubs, $O(N^2)$.

Para terminar, percorre-se o vetor comps procurando o maior tamanho que este contém, O(N) e de seguida o vetor comps é percorrido, e calcula-se a quantidade de vezes que este tamanho se repete, O(N).

Complexidade global da solução: **$O(N^2)$**.

Problema 2- Por razões de simplificação da análise teórica vamos denominar o tamanho do vetor X por N e o tamanho do vetor Y depois do pré-processamento por M, ao invés do nome utilizado no código (size_x e size_y).

Para resolver o problema, começamos por ler os dados de input com ciclos lineares em Θ(N) e Θ(M), e um pré-processamento para o segundo vetor, Y, em O(1) utilizando um unordered map, guardando em Y unicamente os números em comum com o primeiro vetor, X. Seguidamente, é criado um array auxiliar vec_res de tamanho M, em que inicializamos todas as suas entradas a 0, O(M). Aplicando o algoritmo, a ideia é preencher os valores da tabela vec_res e o tamanho atual (cur), percorrendo todos os elementos de X, e para cada elemento de X, percorrer os elementos do vetor Y, O(N*M). Se ambos os vetores tiverem o mesmo elemento, atualizamos o vec_res na posição j para o máximo entre esse mesmo valor e cur + 1 (aumentamos em 1 o tamanho atual), O(1); se for encontrado um tamanho menor em Y, comparativamente a X, atualizamos o valor de cur para o máximo entre o próprio cur e o valor de vec_res na posição j, O(1). Para terminar, vamos percorrer a tabela vec_res procurando o maior tamanho que esta contém, O(M), sendo este o resultado do tamanho que dá resposta ao problema em questão.

Relatório 1º projecto ASA 2021/2022

Grupo: al054
Aluno(s): Diogo Falcão (99199) e Yassir Yassin (100611)

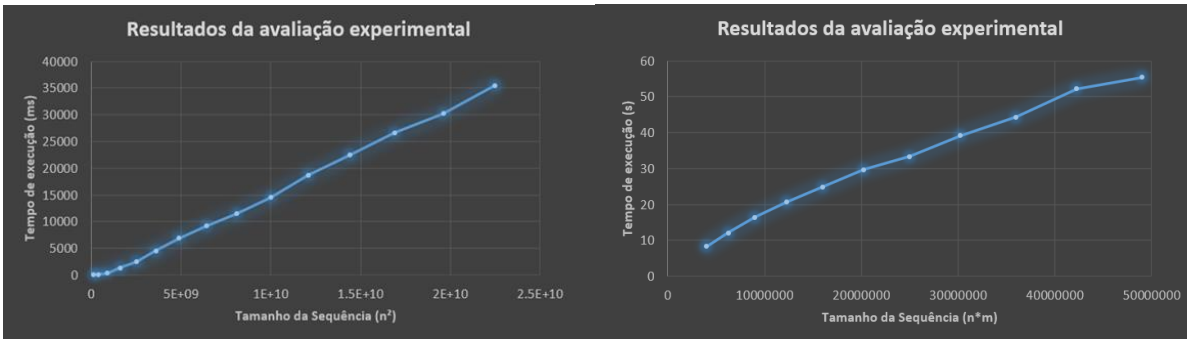
Complexidade global da solução: **O(N*M)**.

Avaliação Experimental dos Resultados

Problema 1- Para calcular o gráfico foram geradas 15 instâncias para sequências de inteiros de tamanho 10000 a 150000 com saltos incrementais de 10000, usando a ferramenta **random_k** fornecida pelo corpo docente para gerar sequências, bem como a std::chrono library para cronometrar o desempenho do algoritmo implementado.

Problema 2- Para calcular o gráfico foram geradas 11 instâncias de sequências de inteiros (com N = M) de tamanhos 2000 a 7000 com saltos incrementais de 500, usando a ferramenta **random_k** fornecida pelo corpo docente para gerar sequências, bem como a std::chrono library para cronometrar o desempenho do algoritmo implementado.

Gráficos:



Problema 1 **Problema 2**

Para o Problema 1, podemos constatar que o algoritmo desenvolvido demonstra a veracidade da complexidade prevista, O(N^2), pelo facto do tempo de execução crescer linearmente com o tamanho da sequência ao quadrado.

O segundo gráfico, não apresenta uma linearidade de forma tão percetível, pelo facto do pré-processamento do segundo vetor de tamanho M poder fazer com que este diminua de tamanho devido ao número de elementos comuns. Apesar disso consegue-se concluir que a complexidade prevista, O(N*M) não está muito longe da obtida.