**Descrição do Problema e da Solução**

Pequena descrição da solução proposta e mapeamento com o problema (1 ou 2 parágrafos).

Problema 1 Dada uma sequência⃗*x* = ⟨*x*0,*x*1,...,*xk*⟩ de inteiros, pretende calcular-se o tama- nho da maior subsequência estritamente crescente de ⃗*x*, bem como o número de subsequências estritamente crescentes de tamanho máximo. Por exemplo, a sequência ⃗*x* = ⟨1, 2, 6, 3, 7⟩ tem duas subsequências estritamente crescentes de tamanho máximo igual a 4: ⃗*s*1 = ⟨1,2,6,7⟩ e

⃗*s* 2 = ⟨ 1 , 2 , 3 , 7 ⟩ .

Problema 2 Dadas duas sequências⃗*x* = ⟨*x*0,*x*1,...,*xk*⟩ e⃗*y* = ⟨*y*0,*y*1,...,*yl*⟩ de inteiros, pretende calcular-se o tamanho da maior subsequência comum estritamente crescente entre ⃗*x* e ⃗*y*, bem como o número de subsequências comuns estritamente crescentes de tamanho máximo. Por exemplo, as sequências ⃗*x* = ⟨1, 2, 6, 3, 7⟩ e ⃗*y* = ⟨1, 2, 4, 7, 3⟩ têm duas subsequências comuns estritamente crescentes de tamanho máximo igual a 3: ⃗*s*1 = ⟨1, 2, 3⟩ e ⃗*s*2 = ⟨1, 2, 7⟩.

Para ambos os problemas para descobrir o tamanho da maior subsequência possível, a ideia é ter um vetor em que na mesma posição de um certo elemento teria o tamanho máximo de uma subsequência crescente que acaba nesse elemento, vamos chamar a esse vetor lensList e para o problema 1 teremos uma lista adicional para quantas subsequências crescentes existem com o tamanho dito na lensList.

Para o problema 1 inicializamos o vetor lensList todo a 1, pois cada elemento pode ser considerado uma subsequência crescente de tamanho 1. Usando uma variável i que irá iterar na lista inicial e uma variável j que para cada i irá percorrer todos elemento da lista com índice menor que i, caso o valor em i seja maior que o valor em j então no lensList se o valor que está em i for menor que valor lensList[j]+1 então o valor em i passa a ser igual ao lensList[j]+1.

Para o problema 2 inicializamos o vetor lensList todo a 0, pois ao contrario do problema 1 neste problema temos de ter atenção se os elementos são comuns as duas listas. À semelhança do problema 1 temos uma variável i que itera em todos os elementos da maior lista e j que para elemento de i ira iterar todos os elementos da segunda lista alem disto teremos também uma variável curr que ira guardar o tamanho da subsequência que estaríamos considerar, esta variável toma o valor 0 a cada valor do i que consideramos.

(É expressamente proibido utilizar fontes externas de código !!)

**Análise Teórica**

Análise teórica da complexidade total e das várias etapas da solução proposta.

* Leitura dos dados de entrada: simples leitura do input, com ciclo(s) a depender linearmente O(n)
* Processamento da instância para fazer alguma coisa. Logo, O(1)
* Aplicação do algoritmo 1 para encontrar a numero e o tamanho das maiores subsequências crescentes numa lista com n elementos. Logo, O(n^2)
* Aplicação do algoritmo 2 para encontrar tamanho da maior subsequência comum crescente entre uma lista com n elementos e outra com m elementos. Logo, O(n\*m)
* Calculo do numero de listas com o maior tamanho possivel. O(n^2)
* Apresentação dos dados. O(1)

Complexidade da solução do 1º algoritmo: O(n^2)

Complexidade da solução do 2º algoritmo: O(n\*m)

**Avaliação Experimental dos Resultados**

Descrição do tipo experiências feitas e gráfico demonstrativo da avaliação de tempos associados.

Gerar pelo menos 10 instâncias (e indicar quais) de tamanho incremental e cálculo dos tempos para cada instância.

Gerar o gráfico do tempo (eixo do YYs) em função do tamanho da instância de entrada (eixo dos XXs) como exemplificado abaixo. Indicar a informação dos eixos.



Concluir se o gráfico gerado está concordante com a análise teórica prevista.