Relatório 1º projeto ASA 2021/2022

Grupo: al004

Aluno(s): Guilherme Leitão (99951) e Sebastião Carvalho (99326)

Descrição do Problema e da Solução

No primeiro problema, é-nos pedido que indiquemos o tamanho da maior subsequência crescente e o número de subsequências com esse tamanho. A nossa solução consiste em criar dois vetores, um para guardar tamanhos e outro para guardar o número de subsequências, e percorrer os elementos do vetor, iterando, em cada elemento, todos os elementos anteriores. Em cada elemento mapeado, o valor da posição no vetor de tamanhos será o tamanho da maior subsequência crescente que contenha o elemento e o valor da posição no vetor de número de subsequências o número de subsequências crescentes máximas com esse elemento. No final do mapeamento, teremos um valor máximo para o tamanho e o respetivo número de subsequências com esse tamanho.

No segundo problema, é-nos solicitado que se calcule o tamanho da maior subsequência estritamente crescente comum a duas sequências de inteiros. Assim, o nosso algoritmo vai ler do input e irá maximizar a eficiência do programa, escolhendo qual das duas sequências é a maior para a iterar apenas uma vez, comparando cada um dos seus elementos a cada elemento da outra sequência. Para além disso, irá apenas considerar um fluxo crescente de elementos, ignorando os elementos da primeira sequência que forem menores que os da segunda. O programa irá representar, através de um vetor com n termos (sendo n o tamanho da menor sequência), o tamanho da maior subsequência estritamente crescente existente entre a maior sequência e a menor até ao n-ésimo elemento, incluindo este obrigatoriamente. O programa também tem em conta a repetição de inteiros ao longo de uma sequência, utilizando lógica adicional e uma variável de controlo. Finalmente, retorna apenas o maior dos elementos do vetor.

Análise Teórica

Problema 1:

- Leitura simples dos dados de entrada, a depender linearmente de n – Θ(n)
- Aplicação da função incSubSeq(): utilizamos um nested loop - O(n^2)
- Apresentação dos dados: é usado printf() - O(1)

Complexidade global da solução: O(n^2)

Problema 2:

- Leitura dos dados de entrada, onde “n” e “m” representam, respetivamente, o tamanho do primeiro e segundo array:

    - 1º Array – depender linearmente de n (aplicação da função readVec()) – Θ(n)
    - 2º Array – Θ[mlog(n)]
- Aplicação da função sizeCommonIncSubSeq(), onde “n” e “m” representam,
- respetivamente, o tamanho do menor e maior array:
    - 1º Nested loop – O(nm)
    - 2ª Loop – O(n)
- Apresentação dos dados: é usado printf() - O(1)

Complexidade global da solução: O(nm)

Avaliação Experimental dos Resultados

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

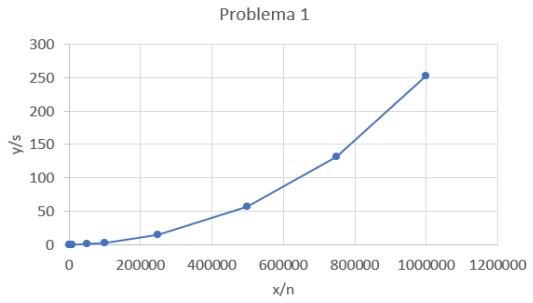
Para o problema 1 e 2 foram utilizados inputs aleatórios de tamanho, respetivamente:

1. 10;1000;5000;10000;50000;100000;250000;500000;750000;1000000.
2. n,m: 10,50;100,500;1000,5000;1500,10000;5000,10000;10000,10000;

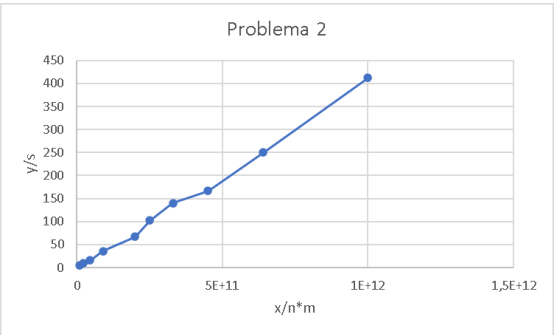
   50000,60000;100000,100000;300000,500000;1000000,1000000.

A tabela seguinte mostra a relação entre o tamanho da(s) sequência(s) de input, e s (segundos), que demora a execução do programa:

x/n	y/s
10	0.006
1000	0.009
5000	0.024
10000	0.036
50000	1.318
100000	2.258
250000	14.81
500000	57.091
750000	131.422
1000000	252.389



x/n	x/m	y/s
100000	100000	4,184
150000	150000	8,964
300000	150000	15,869
300000	300000	35,431
400000	500000	67,22
500000	500000	102,48
600000	550000	140,42
750000	600000	166,27
800000	800000	249,93
1000000	1E+06	412,25



No gráfico do primeiro problema
podemos ver que ele tem uma dispersão quadrática em função de n, e o do segundo tem uma dispersão linear em função n*m. Logo, concluímos que os dados obtidos experimentalmente estão em concordância com os esperados.