Relatório 1º projecto ASA 2021/2022

Grupo: al013
Aluno(s): Diogo Cardoso (99209) e Rafael Oliveira (99311)

Descrição do Problema e da Solução
Para o primeiro problema, que consistia em calcular o tamanho da maior subsequência estritamente crescente (MSEC) de uma sequência, bem como o número de subsequências estritamente crescentes de tamanho máximo, desenvolveu-se uma solução com programação dinâmica utilizando Patience Sort[1] e cálculo/armazenamento do número de caminhos que cada elemento da sequência (representado por uma carta) pode prolongar que levam a uma MSEC. A fórmula do cálculo dos caminhos segue o princípio de que uma carta só prolonga um caminho se for maior que a carta anterior nesse caminho e que, sendo que é sempre menor que todas as cartas da pilha onde está a ser inserida, todas as MSEC a acabar no topo desta pilha podem acabar nela.

Quanto ao segundo problema, que consistia em calcular o tamanho da maior subsequência comum estritamente crescente entre duas sequências, também foi utilizada uma solução com programação dinâmica. No processamento do *input*, apenas são guardados os valores da segunda sequência comuns à primeira de modo a poupar processamento na resolução do problema. Na resolução, é criada uma tabela com o comprimento de um dos vetores e em cada índice é guardado o tamanho da MSECC que acaba em cada índice do vetor escolhido, respetivamente, com base nas seguintes regras[2]:

- Para cada elemento do primeiro vetor, percorrer todos os do segundo.
- Quando se encontra um elemento maior, a variável que guarda o tamanho da MSECC atual é atualizada (maior valor entre o tamanho atual e o valor encontrado na tabela auxiliar no índice atual).
- Quando se descobre um elemento igual, a tabela auxiliar é atualizada (maior valor entre o que se encontra atualmente na tabela ou tamanho da MSECC atual + 1).
- A variável que guarda o tamanho da maior MSECC é atualizada caso o último valor calculado seja maior que o atual.

Análise Teórica
Problema 1:

- Leitura dos dados de entrada: simples leitura do input, com ciclo a depender linearmente de *n* (tamanho do vetor *V*). *Θ(n)*
- Para cada elemento de *V*, aplicação do algoritmo *upperbound* de modo a encontrar a pilha onde colocar o elemento. *O(logn)*
- Adição do elemento à pilha correspondente e cálculo da soma do mesmo com o algoritmo *get_sum* (binary search). *O(logn)*
- Apresentação dos dados. *O(1)*

Complexidade temporal global da solução: *O(nlogn)*
Complexidade espacial global da solução: *O(n)*, pois apenas se mantêm em memória as cartas.

Relatório 1º projecto ASA 2021/2022

Grupo: al013
Aluno(s): Diogo Cardoso (99209) e Rafael Oliveira (99311)

Problema 2:

- Leitura dos dados de entrada: leitura e pré-processamento do input que depende linearmente de n e m (tamanhos de ambos os vetores). No pior caso são processados todos os elementos. *O(nm)*
- Para cada elemento do primeiro vetor, percorrer todos os elementos do segundo vetor e atualização da array de apoio e da variável que guarda o maior tamanho. *O(nm)*
- Apresentação dos dados. *O(1)*

Complexidade temporal global da solução: *O(nm)*
Complexidade espacial global da solução: *O(n + m)*, pois apenas se guardam as sequências, uma delas uma vez e a outra duas vezes.

Avaliação Experimental dos Resultados
Foram gerados *inputs* de tamanho variável com o pior caso (números consecutivos e vetores iguais para o problema 2) para cada problema e medido o tempo de resposta, de acordo com o demonstrado nos gráficos abaixo.
Para o problema 1, foram utilizados os valores nos intervalos [10, 99, 1], [100, 999, 10], [1000, 9999, 100], [10000, 99999, 1000], [100000, 999999, 10000] e [1000000, 9999999, 100000] resultando num total de 541 *datapoints.* Para o problema 2 foram utilizados os mesmos valores com exceção dos dois últimos intervalos, resultando num total de 360 *datapoints*.

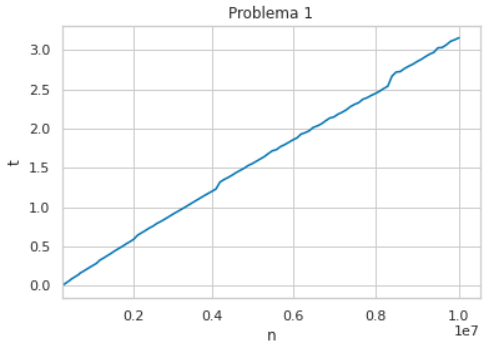

Fig. 1: Tempo de resposta ao problema 1 (em segundos) em função do tamanho do input, este em escala nlogn.

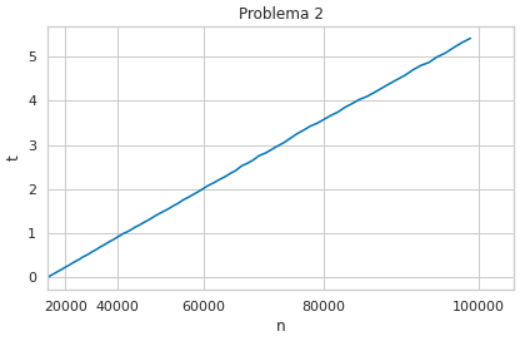

*Fig. 2: Tempo de resposta ao problema 2 (em segundos) em função do tamanho do input (*n = m*), em escala* nm.

A relação linear em ambos os gráficos prova a complexidade temporal de pior caso esperada para os problemas: *O(nlogn)* e *O(nm)*, respetivamente.

Referências
[1] Burstein, A.; Lankham, I. (2006). *Combinatronics of patience sorting piles.*
[2] Sakai, Y. (2006). *A linear space algorithm for computing a longest common increasing subsequence.*