Relatório 1º projecto ASA 2021/2022

Grupo: al030

Aluno(s): Guilherme Almeida Patrão (99230) e Inês Ye Ji (99238)

Descrição do Problema e da Solução

Sendo P1: Tamanho máximo das subsequências estritamente crescentes de uma sequência e o número existente delas e P2: Tamanho máximo das subsequências comuns estritamente crescentes de duas sequências, as nossas soluções para ambos foram através da aplicação de algoritmos que recorrem ao uso de DP (programação dinâmica) e ao uso de tabelas para guardar os valores durante as iterações.

Análise Teórica

Na leitura de dados, utilizamos o `std::getline()` para ler o input. Depois, procedemos à transformação da string de input em ints (O(n)), adicionando-os ao vetor da sequência correspondente (tempo de inserção nas estruturas O(1)). No caso do problema 2 comparamos entre os elementos das sequências 1 e 2 de modo a que a sequência 2 apenas tenha elementos que existem na 1 (utilizou-se `std::unordered_map` com tempo de procura O(1) na maioria dos casos, O(n) na pior das hipóteses). Logo, a complexidade é O(n).

No problema 1 temos as seguintes tabelas: uma tabela para o tamanho com tamanho NxN e uma para o nº de subsequências (counter) de 1xN (inicializadas a 1, N = nº de elementos da sequência, j refere-se ao número da linha, i ao da coluna). Na 1ª tabela, preenche-se de linha a linha e o número de colunas que é preenchido por linha é de 1 a n-1 (começando na 2ª linha), pois são a quantidade de números a que o número da respetiva linha é comparado. Preenche-se da esquerda para a direita, e caso se verifique que o elemento anterior na sequência (ou seja na linha anterior) é menor que o elemento que estamos agora a avaliar e menor que o maior número existente na subsequência de maior tamanho, então podemos diretamente preencher na última coluna dessa linha o tamanho máximo daquele momento + 1 e o counter continua com o valor do índice anterior. Caso contrário, comparamos o número da linha com cada um da coluna que vem antes deste na sequência e se este for menor e se o último valor preenchido da linha com igual número à da coluna a ser comparada [i] for igual ou maior que o valor a preencher [j], então o valor de [j] é substituído pelo valor [i] + 1 e o counter do número [j] é substituído pelo do [i]. Caso o último valor preenchido [i] + 1 seja igual a [j], então estamos num caso em que há mais que uma sequência desse tamanho, logo adiciona-se ao counter desse número [j] o do número a que se comparou [i].

O maior tamanho é o maior valor dos últimos valores preenchidos por linha (valores na diagonal a partir da segunda linha) e o maior número de tais subsequências é a soma de todos os counters cujos números a que pertencem são também os números que indicam a subsequência de maior tamanho. Para

Relatório 1º projecto ASA 2021/2022

Grupo: al030

Aluno(s): Guilherme Almeida Patrão (99230) e Inês Ye Ji (99238)

---

calcular o valor de cada célula, a complexidade é O(1). Logo, a complexidade é Ω(n) (Ω(n) * Ω(1)) (sequência estritamente crescente) e $O(n^2)$ ($O(n^2)$ * O(1)) (maior parte dos casos).

A tabela do problema 2 é de tamanho NxM (N = nº de elementos da sequência 1 e linhas da tabela; M = nº de elementos da sequência 2 e colunas da tabela). Preenche-se a tabela de linha a linha (inicializada a 0), se N = M e o valor do length (tamanho atual, inicializa-se a 0 sempre que se muda de linha) for igual ou maior que o número a preencher, então é preenchido com length + 1, isto é, encontrou-se um número em comum crescente. Caso N > M, e o length for menor que o número a preencher, então o length é atualizado para esse valor. Este algoritmo apenas guarda a última linha da tabela, pois cada linha corresponde a uma iteração do j. Assim, podemos concluir o resultado através do maior número na última linha da tabela. Para percorrer a tabela, a complexidade é O(NxM) e para calcular cada célula é O(1). Logo, podemos concluir que a complexidade geral do algoritmo é O(NxM).

Avaliação Experimental dos Resultados

Foram realizados 10 testes, em que a cada teste incrementou-se o tamanho das sequências do input por 100 mil (de 100 mil até 1 milhão). Este foi realizado através do gerador de instâncias fornecido (ex: ./random_k 1 10 0.99 1000000 (problema 1) e ./random_k 2 10 0.99 1000000 1000000 (problema 2)). Os gráficos traduzem a duração em segundos (YYs) de execução do algoritmo do problema 1 e 2 para uma sequência com X nº de elementos (XXs).

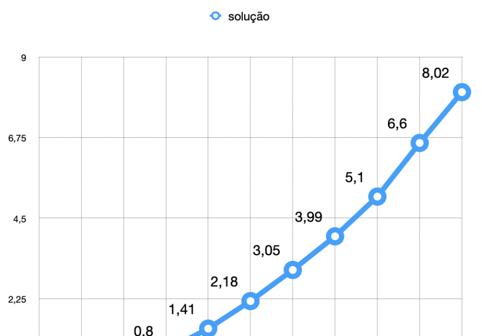

Gráfico 1 - Problema 1

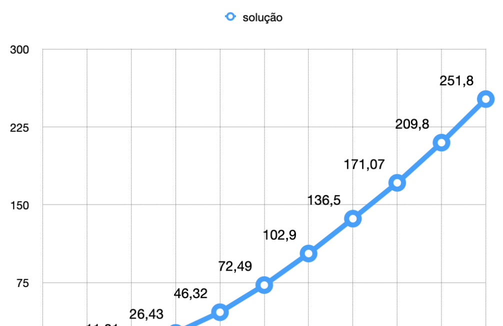

Gráfico 2 - Problema 2

A partir do gráfico 1 podemos verificar que o algoritmo utilizado é de complexidade $O(n^2)$ pois a curva é metade de uma função quadrática. No gráfico 2 observamos que a curva é parecida a uma função quadrática mas não o chega a ser pois a complexidade do algoritmo é O(nm) e m nem sempre é do mesmo tamanho que n (devido ao pré-processamento do input).

Relatório 1º projecto ASA 2021/2022

Grupo: al030

Aluno(s): Guilherme Almeida Patrão (99230) e Inês Ye Ji (99238)

Referências: https://youtu.be/fV-TF4OvZpk / https://iq.opengenus.org/longest-common-increasing-subsequence/ / Notas do professor.