Relatório 1º projecto ASA 2021/2022

Grupo: al003

Aluno(s): Renato Tito Antunes de Meireles Martins (102314) e Luís Filipe Pedro Marques (99265)

Descrição do Problema e da Solução

Dada uma sequência de inteiros, pretende-se calcular o tamanho da maior subsequência estritamente crescente desta sequência, bem como o número de subsequências estritamente crescentes de tamanho máximo. São inicializados dois vetores do tamanho do vetor inicial com todos os elementos a 1, que vão sendo modificados ao longo da iteração pelo vetor, sendo usados para determinar a LIS[1] e o seu número de ocorrências naquele índice, respetivamente. No final da execução do programa, teremos a máxima LIS[1] da sequência original e o seu respetivo número de ocorrências, graças ao uso de variáveis que servem de acumuladores que vão sendo incrementadas até obter o resultado final.

Análise Teórica

- Leitura dos dados de entrada: leitura do input na função *readVector*, guardando os valores inseridos num vetor, com recurso a um ciclo *for* dependendo linearmente de *V*(número de elementos do vetor). Logo, a complexidade é O(V).

- Aplicação do algoritmo para calcular o número de subsequências estritamente crescentes de tamanho máximo. A complexidade é O(n^2), pois dentro da função *maxNSubSeq* são executados dois ciclos *for* um dentro do outro e cada um com complexidade O(n).

- A complexidade da apresentação dos dados é O(1).

Sendo assim, a complexidade temporal global da solução é O(n^2) e a complexidade espacial é O(n), pois depende linearmente do tamanho do input, sendo que a resposta é guardada em N sub problemas.

Descrição do Problema e da Solução

Dadas duas sequências de inteiros, pretende calcular-se apenas o tamanho da maior subsequência comum estritamente crescente entre ambas. É lido o primeiro vetor sendo que os seus elementos serão colocados num *map* cuja *key* será os inputs lidos e a partir desse map quando for lido o vetor 2 só serão colocados elementos comuns entre ambos os vetores noutro vetor. No vetor 1 são removidos os elementos não comuns ao vetor 2.

É inicializado um vetor que regista as LIS[1] por índice. São percorridos ambos os vetores e em cada iteração do primeiro vetor é percorrido o segundo por completo com o auxílio de um contador de LIS[1] e de um registo desse contador por índice (determinar a LIS[1] comum do início até esse índice) sendo esse registo incrementado no índice onde existem elementos em comum do 2º vetor com o elemento do 1 º vetor a ser percorrido e que ainda não tenha sido contabilizado. Caso se verifique que existe uma relação decrescente entre o elemento do 1º vetor e os elementos do 2º vetor, significa que falta contabilizar esses elementos na LIS[1] a determinar. Assim, o contador é atualizado com o elemento do registo de índice relativo ao índice a ser percorrido no 2º vetor de forma que o contador esteja bem atualizado. No fim da iteração do segundo vetor o contador é retornado a zero para a recontagem no próximo índice

1: Longest Increasing Subsequence

Relatório 1º projecto ASA 2021/2022

Grupo: al003

Aluno(s): Renato Tito Antunes de Meireles Martins (102314) e Luís Filipe Pedro Marques (99265)

---

do 1ºvetor. No fim, determina-se o máximo desse registo devolvendo o elemento do índice onde o contador, ou seja, a LIS[1], é superior.

Análise Teórica

- Leitura dos dados de entrada: leitura do input na função *readVector*, guardando os valores inseridos num vetor, com recurso a um ciclo *for* dependendo linearmente de *V*(número de elementos do vetor). Logo, a complexidade é O(V).

- Pré-processamento do input onde é devolvido um vetor apenas com os elementos comuns entre ambas as sequências. É feita esta procura de elementos iguais em *unordered maps,* isto é, utilizando a função *find,* permitindo ter uma complexidade de O(1). Se esta operação fosse aplicada em vetores, a sua complexidade seria O(n).

- Caso o 2º vetor tenha elementos não comuns removemos do 1º vetor os que não são comuns numa função O(n^2) pois percorre os dois vetores.

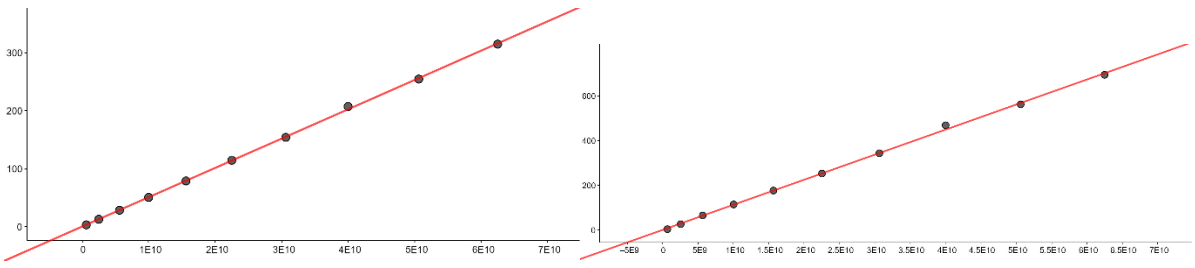
- Aplicação do algoritmo para calcular o tamanho da maior subsequência crescente entre ambas as sequências iniciais, cuja complexidade é O(n^2), dado que são utilizados dois ciclos *for,* um dentro do outro e cada um destes com complexidade O(n).

- No fim teremos um vetor com as LIS[1] comuns por índice e usando uma função que percorre todo o vetor de registo das LIS´s[1] O(n) e compara o seu valor determinando o seu máximo.

- A complexidade da apresentação dos dados é O(1).

Avaliação Experimental dos Resultados

Ambos os exercícios foram executados com vetores aleatórios de tamanho compreendido entre 25 000 e 250 000 elementos. O tamanho dos vetores ao quadrado e o respetivo tempo de execução para o problema 1 e para o problema 2, respetivamente, está representado nos gráficos a seguir:



Os valores no eixo dos *yy* dizem respeito ao tempo de execução e os valores no eixo dos *xx* referem-se ao quadrado do tamanho do input. Note-se que foram utilizados os quadrados dos valores do input no eixo dos *xx* devido à previsão que o algoritmo em ambos os problemas é de complexidade O(n^2), algo que foi comprovado experimentalmente com a obtenção de uma reta crescente em ambos os gráficos.

1: Longest Increasing Subsequence

Relatório 1º projecto ASA 2021/2022

Grupo: al003

Aluno(s): Renato Tito Antunes de Meireles Martins (102314) e Luís Filipe Pedro Marques (99265)

Pseudocódigo exercício 2

Devido ao facto de este não ser um algoritmo clássico, coloca-se aqui o pseudocódigo:

1- Ler o vetor 1 e colocá-lo num unordered map onde as chaves são os elementos e em cada elemento é colocado um valor 1.
2- Ler o vetor 2 e só colocar no vetor 2 os elementos cuja key existe no unordered map referido.
3- Remover os elementos não comuns entre os dois vetores no vetor 1.
4- Inicializar um vetor que regista a LIS no índice em que está localizado sendo este do tamanho do 2º vetor.
5- Percorrer o vetor 1 alterado (vetor 3) usando um ciclo *for* e em cada interação(i) é inicializado uma variável a zero chamada *resultado_corrente*.
6- Dentro do ciclo *for* percorremos o segundo vetor usando outro ciclo *for* e em cada interação (j) comparamos os valores existentes nos respetivos índices (i,j) ,sendo que caso sejam iguais associamos e incrementamos o *resultado_corrente* ao registo de *resultados_correntes* no índice j. Caso contrário, se o valor no vetor 1 no índice i é superior ao valor no vetor 2 no índice j e se o valor do índice j no registo for superior ao *resultado_corrente* guardado nessa interação, este é atualizado com o registo do *resultado_corrente* no índice j.
7- No fim, procuramos o valor máximo no vetor onde estão registados os *resultados_correntes* por índice usando uma função que percorre o vetor e compara elementos com auxílio de uma variável onde será guardada o máximo, sendo retornado este valor.

1: Longest Increasing Subsequence