Relatório 1º projecto ASA 2021/2022

Grupo: tp063

Aluno(s): João Vaz (98946) e André Santos (99730)

Descrição do Problema e da Solução

- Com o objetivo de calcular a **maior subsequência crescente** de um array **"x[]"** de tipo inteiro e tamanho **"n"**, criamos dois arrays **"lengthSC[]**" e **"countSC[]**" do tipo inteiro de tamanho **"n"**, em que **"lengthSC[]**" irá guardar o tamanho da maior subsequência crescente na posição correspondente de **"x"** e **"countSC[]**" irá guardar o número de maiores subsequências crescentes na posição correspondente de **"x".** Devolve uma estrutura com o tamanho da **maior subsequência crescente** e as vezes que esta aparece.
- Para calcular a **maior subsequência comum crescente** entre dois arrays **"x[n]"** e **"y[m]"**, inicializamos um array de 0 's, **"res[m]".** Para cada elemento de x percorremos o array y, procurando elementos comuns. Se este for encontrado e o valor for superior guardamos o tamanho novo de uma LCIS na respectiva posição j.

Análise Teórica

MSC

- Com o objetivo de calcular a **maior subsequência crescente** de um array **"x[]"** de tipo inteiro e tamanho **"n"**, criamos dois arrays **"lengthSC[]**" e **"countSC[]**" do tipo inteiro de tamanho **"n"**, em que **"lengthSC[]**" irá guardar o tamanho da maior subsequência crescente na posição correspondente de **"x"** e **"countSC[]**" irá guardar o número de maiores subsequências crescentes na posição correspondente de **"x".**Devolve uma estrutura com o tamanho da **maior subsequência crescente** e as vezes que este aparece.
- O array **"lengthSC**" irá guardar o número da maior subsequência crescente na posição correspondente de **"x".**
- O array **"countSC**" irá guardar o número de maiores subsequências crescentes na posição correspondente de **"x".**
- Criamos 2 loops, onde o loop exterior irá iterar x de i =1 -> i =n-1 e o loop interior irá iterar entre j = 0 -> j = i-1.
- Verificamos se o valor na posição **i** do array **"x[]"** é maior do que o valor na posição **j**. Se for verdade, verificamos se o valor+1 na posição **j** no array **"lengthSC**" é maior do que o valor na posição i, se for verdade atualizamos o valor
- guardar o maior aumento de subsequências crescentes(em cada índice de x), iterando pelos elementos do array recebido, retornando o maior valor(O(n)) e armazenar o número do conjunto mais longo de subsequências crescentes (em cada índice de x), portanto O(n^2).

Relatório 1º projecto ASA 2021/2022

Grupo: tp063

Aluno(s): João Vaz (98946) e André Santos (99730)

MSCC

- **Leitura dos dados de entrada**:
- 1.Leitura de dados para um array “x” (Exercício 1):
- x = malloc((r=100) *int)
- Input(value, character)
- While (character!=”ChangeOfLine”) -> Enquanto não se mudar de linha
    - Input(x[i] = value, character)
    - i++
    - If i+1>r -> Temos que alocar mais espaço para ler mais dados
        - r = r*2
        - x= realloc(r*int)
    - End If
- End While
- Ciclo dependente de i, logo terá uma complexidade de O(n).
-
- 2. Leitura de dados para um array “x” e um array “y” (Exercício 2):
- Igual à leitura de dados do primeiro processo, mas realizado 2 vezes O(n)+O(n), com loops while de evolução l1 ++ e l2 ++ correspondentemente.
- **Processamento da instância:**
- Objetivo:
    - Retirar elementos não comuns de cada array.

```
arr = x
count=0
arr = qsort(arr)
For i = 0 to Size_Of_Array
        Int Key = x[i]
        Ip = bsearch(key,arr)
        If (ip exist)
                y[count++]=key
        EndIf
End For
```

- Fazemos uma cópia ordenada (O(nlog(n)) de cada array (x e y) e de seguida, realizamos uma pesquisa binária (O(log(n))) de modo a encontrar os elementos comuns. Ao encontrar colocamos os elementos na posição count, que irá ser incrementado em 1 valor. Logo, O(nlog(n)).

Relatório 1º projecto ASA 2021/2022

Grupo: tp063

Aluno(s): João Vaz (98946) e André Santos (99730)

Com o objetivo de calcular o número da maior subsequência crescente comum entre dois arrays “x[]” e “y[]” de tipo inteiro e tamanho “m”, criamos um array “res[]” em que “res[]” será preenchido com 0´s e com o mesmo tamanho que “y[]”.
De seguida, criamos uma variável “aux” inicializada em 0. Para cada “x[i]”, iteramos pelos elementos de “y[]” com o objetivo de encontrar elementos em comum nos dois arrays, portanto um “x[i]” == “y[j]”.
Caso seja encontrado um elemento em comum entre os dois arrays recebidos, irá ser incrementada a variável “aux” e armazenado o valor em “res[j]”.
Por outro lado, se o valor de “x[i]” > “y[j]”, a variável “aux” irá ser atualizada com o valor máximo que esteja em “res[j]”.
Terminando, assim, com o nosso objetivo de retornar o valor da maior subsequência comum entre os dois arrays recebidos, neste caso, o maior valor guardado em “res” como resultado, aplicando a função “biggestElement()” , função essa que itera pelos elementos à procura do maior valor, guardando-o em “res”.

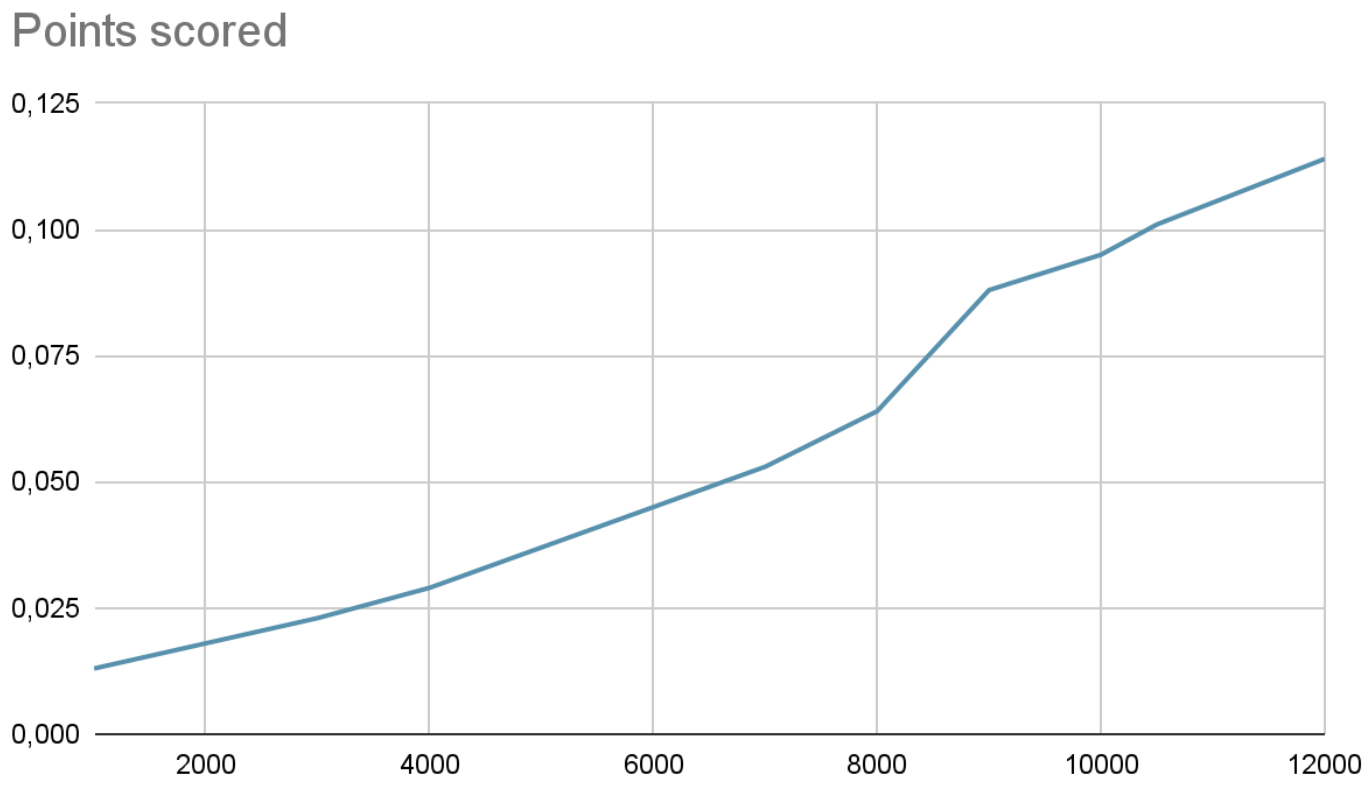
Complexidade global da solução: O(n^2).

Relatório 1º projecto ASA 2021/2022

Grupo: tp063

Aluno(s): João Vaz (98946) e André Santos (99730)

Avaliação Experimental dos Resultados



O gráfico gerado está concordante com a análise teórica prevista.