Trabalho Prático 3
Exposição de tecidos

João Vitor Ferreira

Universidade Federal de Minas Gerais (UFMG)

Belo Horizonte — MG — Brasil

joaovitorferreira@ufmg.br

1. Introdução

Este arquivo visa documentar e analisar o funcionamento do programa “Loja”. Ele implementa um código em que realiza a computação do “Longest Decreasing Subsequence” de um dado vetor. Para este trabalho tem se que, este vetor contém rolos de tecido, correspondente a remessa recebida e ele deve dispor os rolos de forma que eles estejam de maneira decrescente em preço e consiga-se encontrar a solução onde a soma dos valores de cada tecido seja a máxima decrescentemente, inserido apenas no lado esquerdo ou direito da prateleira.

2. Método

O programa foi desenvolvido na linguagem C++, compilada pelo compilador G++ da GNU Compiler Collection.

2.1. Algoritmo

Para a implementação deste problema foi realizado a utilização e implementação de um algoritmo conhecido como “Longest Decreasing Subsequence”, uma variação do algoritmo visto em sala de aula, “Longest Increasing Subsequence”, ou, em português “Maior Subsequência Descendente” e “Maior Subsequência Ascendente” respectivamente. Este algoritmo consiste em encontrar o tamanho da maior

subsequência decrescente terminada com o i-ésimo elemento, mas para o intuito deste trabalho algumas alterações tiveram de serem feitas para que o algoritmo encontrasse o resultado esperado. Porém, as alterações foram ao método de inserção e não no algoritmo em si que permaneceu intacto em sua estrutura.

Originalmente o algoritmo em si, possui custo $\theta(n^2)$. A ideia é criar um vetor que armazenará o comprimento do LDS terminando em cada índice. Depois, iterar através pelo vetor para encontrar o maior LDS e adicionar a este comprimento para encontrar o LDS terminando no índice atual.

A implementação do algoritmo usado foi o pseudocódigo passado em sala de aula pelo professor Vinicius. Com algumas alterações para o funcionamento do mesmo para este problema:

- Como o algoritmo passado foi o LIS, e queremos o LDS, apenas multiplicamos todos os elementos do vetor por -1
- Devido à forma que foi proposto o problema, a inserção teve de ser alterada. Como podemos inserir tanto no começo e fim do vetor, ao ler os dados realizamos essa inserção tanto no início quanto no fim, para que o algoritmo consiga encontrar e produzir uma saída satisfatória.

Vale ressaltar que grande parte da eficiência da solução adotada vem de ter sido requisitada e implementada a partir dos princípios de Programação Dinâmica. Que proporcionaram um custo menor por encontrar a solução para problemas menores até que se encontre a solução para o problema desejado.

2.2. Relação de Recorrência

Como estamos implementando um algoritmo de Programação Dinâmica, um dado extremamente importante de se apresentar é a relação de recorrência do algoritmo implementado. Para este sendo implementado ela é:

$$LDS(i)\begin{cases} 1, & em\ caso\ de\ haver\ somente\ 1\ rolo\ na\ prateleira \\ \max(1, lds[i]+1), & para\ i=0\ até\ n-1\ e\ para\ cada\ rolo[i]\ a\ fim\ de\ encontrar\ um\ custo\ maior\ que\ o\ n-ésimo\ custo. \end{cases}$$

2.3. Análise de Complexidade

Como definido na documentação, deveríamos obrigatoriamente usar um algoritmo que possua custo.$\theta(n^2)$ E isso foi cumprido com o algoritmo usado, para comprovar isto analisaremos a complexidade dele:

Temos o primeiro loop que realiza inicialização do vetor lds com 1 e depois em um loop aninhado realiza a definição do valor como lds[j]=lds[i] +1 se o lds[j] for maior lds[i] e o mesmo caso para rolos. Logos temos complexidade.$\theta(n^2)$

No segundo loop que realiza a computação do maior valor temos apenas um if nele. Portanto, a complexidade é $\theta(n)$.

Com isso a complexidade total do algoritmo é:

$$\theta(LDS) = \theta(n^2) + \theta(n)$$

$$\theta(LDS) = \theta(n^2)$$

2.4. Classes

Para a resolução deste trabalho foi feita a criação de uma classe Loja que armazena o vetor de rolos e possui a função de inserção e cálculo de LDS.

3. Estratégias de Robustez

Para garantir o correto funcionamento do código foram adicionados errosAssert, dos quais param o código toda vez que um erro é localizado. Os erros adicionados estão relacionados com os valores de entrada. Tais são o número de teste que deve ser 1 ≤ N ≤ 10, a quantidade de rolos 1 ≤ R ≤ 20000 e o valor do rolo 1 ≤ P ≤ 1000000