Trabalho Prático 2

Avaliação do evento

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 “Rock”. Ele implementa um código em que realiza o cálculo da localização onde está contida o subvetor de soma máxima de dado vetor. Este vetor que contêm notas atribuídas por amigos a cada show do evento. E deseja-se encontrar o SSM que tenha as maiores notas.

2. Método

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

2.1. Algoritmos

Para a implementação deste problema foi realizado a utilização e implementação de um algoritmo conhecido como subvetor de soma máxima, ou, Max crossing subarray. Este algoritmo consiste em encontrar a maior some consecutiva de elementos originalmente, mas para o intuito deste trabalho algumas alterações tiveram

de serem feitas para que o algoritmos não somente encontrasse a soma máxima e sim o segmento dele que a contém, com os índices iniciais e finais dos elementos que compõe essa soma.

Originalmente o algoritmo em si, possui custo $\theta(n \log n)$ e depois das modificações performadas para atingir o objetivo seu custo manteve inalterado, pois todas as alterações feitas no SSM (nome da função que contém o algoritmo), possuem custo,$O(n)$ logo são lineares, o que não irá afetar esta complexidade.

A implementação do algoritmo teve como base o pseudocódigo passado em sala de aula, as mudanças realizadas foram 4:

- A primeira mudança foi a criação de um vetor de posição para as notas em que sempre onde a chamada recursiva chega à condição de parada esta posição é salva no vetor.
- A segunda alteração foi criação de uma função que analisa o vetor de notas e retorna à posição em que houve um empate, logo, onde a soma das notas é igual a 0. Caso essa posição não exista é retornada uma posição invalida, para isso foi escolhida a posição -100001
- A terceira a mudança está ligada a primeira que foi a inserção dessa posição retornada no vetor de posição das notas. Ela é somente inserida caso ela difira do valor de posição invalida.
- Por fim, a última mudança foi a criação de condições para poder saber que posição o SSM está se é na direita, esquerda ou no cruzamento. Após isso, o vetor de posição das notas e notas empatadas é carregado para o vetor da classe solução.

Vale ressaltar que grande parte da eficiência da solução adotada vem de ter sido requisitada e implementada a partir dos princípios de divisão e conquista. Que proporcionaram um custo menor e complexidade de legibilidade menor do que uma implementação por meios tradicionais e iterativos, sem o uso da recursão.

2.2. Classes

Para a resolução deste trabalho foi feita a criação de uma classe Rock que armazena os dados e funções necessários para este programa.

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 da quantidade de amigos e shows. No caso de se ter menos de 1 amigo ou show ou se ter mais de 50 amigos ou 100000 shows o programa é abortado, e deve-se adequar a entrada para que ele possa ser executado corretamente.

4. Conclusão

Após a implementação do programa foi notório que ele pode ser visto que o uso de divisão e conquista é crucial para o bom desempenho da execução deste programa, pois garante que para entradas maiores do que o usual o uso de recursos não seja alarmante e um motivo de gargalos e a não execução do código.

Outro fator interessante que se pode tirar conclusões sobre é o fato do algoritmo de SSM ser extremamente versátil e poder ser levemente adaptado sem que haja grandes impactos para seu custo e complexidade, já que as alterações realizadas para este projeto tiveram baixo impacto no desempenho do programa como pontudo acima.