Trabalho Prático **02**

Documentação

DCC206 Algoritmos 1

Universidade Federal de Minas Gerais ***2022/02***
Bernardo Reis de Almeida *(2021032234)*

1. Introdução

O presente trabalho prático tem por objetivo o exercício da **modelagem computacional de problemas** orientada ao paradigma de programação **divisão e conquista**. Em particular, dada uma situação problema real, tem-se em vista a **abstração** e a **implementação** desta em uma **estrutura de dados** imbuída do paradigma mencionado e que apresente uma metodologia de solução correta e eficiente para a questão, a qual é descrita a seguir.

Um grupo de amigos irá participar do festival musical Rock In Rio. O festival é organizado em vários concertos que ocorrem de maneira sequencial e individual, isto é, uma banda ocupa o único palco por vez e cada uma se apresenta após a outra. Todo o ano, as mesmas bandas se apresentam na mesma ordem. Interessados em extrair o maior proveito possível do evento, os amigos farão o uso de um aplicativo para atribuir notas a cada um dos shows. Cada indivíduo deverá avaliar cada show em uma escala de -5 a +5. Ao final do festival, com as notas de todos em mãos, estas serão somadas e **o grupo irá escolher quais bandas assistir no próximo ano de modo a satisfazer ao máximo cada integrante**, levando em conta que uma vez fora do evento, não é possível retornar.

Em mais detalhes, o programa irá receber como entrada a quantidade de amigos no grupo, a quantidade de shows no evento e as notas atribuídas por cada integrante a cada apresentação. A partir destes dados, deve ser retornado **o intervalo de shows a ser acompanhado de modo a maximizar a satisfação do grupo como um todo**, no formato **[índice_do_primeiro_show índice_do_último_show]**. A presente implementação foi realizada na linguagem **C++** - o compilador utilizado foi o **G++** - e em uma plataforma **Unix**.

Inicialmente, a abstração do problema em um modelo computacional será abordada na seção **[Modelagem]**, na qual as estruturas de dados utilizadas e as estratégias escolhidas serão detalhadas. Em seguida, na seção **[Execução & Funcionamento]**, o fluxo de trabalho do algoritmo será discutido, relevando a maneira como aquilo definido na seção anterior é efetivamente utilizado na resolução da questão. Por fim, na seção **[Considerações Finais]**, há instruções para se compilar e para se executar o programa, além de referências bibliográficas para todas as fontes de consulta utilizadas.

2. Modelagem

O **aplicativo de avaliações**, sendo este o principal elemento da situação problema enunciada, foi modelado como um **tipo abstrato de dados** - e implementado por meio de uma **classe** na linguagem C++. Suas **características** e **funcionalidades** - ou **atributos** e **métodos** - buscam abstrair de maneira completa e precisa a realidade com a qual se trabalha.

O **número de integrantes** no grupo de amigos e o **número de shows** no festival de música foram ambos representados por **variáveis simples**. As **avaliações** de cada membro para cada apresentação foram organizadas em uma **matriz** de tamanho **[número de amigos] x [número de shows]**. Tal escolha é um resultado da própria natureza dessa fonte de dados: para cada amigo, haverá uma quantidade de avaliações equivalente ao número de apresentações no evento, de modo que cada entrada da matriz seja devidamente fornecida e inicializada, o que justifica o investimento em memória e é recompensado pelo acesso direto a qualquer elemento. A partir dessa matriz, ainda, um **vetor** de tamanho **[número de shows]** também é inicializado e contém simplesmente a soma das avaliações de cada amigo por show.

Nesse momento, define-se que a **satisfação com relação às apresentações** é dada pela **soma de suas notas gerais**, as quais, por sua vez, são a soma das notas individuais de cada amigo. Além disso, **caso existam dois intervalos de shows empatados em termos de satisfação, o critério seguinte de desempate é o seu tamanho**: o escolhido será aquele com mais apresentações. **Caso ainda assim haja empate, o intervalo escolhido será aquele que ocorre em um horário mais cedo**. Outro detalhe importante é que as informações providas como entrada já estarão ordenadas, isto é, para cada linha (referente a uma pessoa), o primeiro elemento (nota) é relativo ao primeiro show, o segundo, ao segundo show e assim por diante, em ordem crescente de horário na programação do festival musical.

	1° Show	**2° Show**	**3° Show**
1° Amigo	1	2	3
2° Amigo	4	-5	1
3° Amigo	-2	-1	6
Total	3	-4	10

Satisfação com o intervalo de shows [1, 3] = (3) + (-4) + (10) = **9**

Dadas essas informações, o problema de se definir o intervalo de apresentações que maximiza a satisfação do grupo pode ser pensado como **encontrar o intervalo sequencial de elementos do vetor declarado que maximiza sua soma**, isto é, o subvetor de soma máxima.

A solução designada é baseada no paradigma de **divisão e conquista**, o qual enuncia a divisão de um **problema maior** em **subproblemas menores**, o solucionamento individual destes e sua combinação para a obtenção de uma resposta à questão geral. Esta, no caso, foi definida como encontrar o subvetor que maximiza a soma de seu conjunto sequencial de elementos. Aqui, observa-se que há **três possibilidades** mutuamente exclusivas: ou ele está na partição esquerda do vetor original, ou está na partição direita, ou cruza o ponto mediano dessa estrutura de dados. Em outras palavras, encontrar esse subvetor equivale a determinar o maior subvetor na partição esquerda, o maior subvetor na partição direita, o maior subvetor que passa pelo ponto mediano e retornar o maior dentre eles. Naturalmente, esses três **subproblemas** corresponderão à divisão da questão original e cada solução será utilizada para a conquistar.

Vetor Original

1 2 3 4 5 6 7 8 9 0

Possibilidades

1 **[2 3 4]** 5 || 6 7 8 9 0 1 2 **[3 4 5** || **6 7 8]** 9 0 1 2 3 4 5 || 6 **[7 8 9]** 0

Um detalhe a ser mencionado é que, para os fins do presente trabalho, foi definido que o "meio" de um vetor com uma quantidade par de elementos é a transição que o divide em duas metades de mesmo tamanho, enquanto que, em um vetor com uma quantidade ímpar de elementos, é a transição do elemento mediano para o seu vizinho direito.

Vetor Par	**Vetor Ímpar**
1 2 || 3 4	1 2 3 || 4 5

Em termos de implementação, o algoritmo segue uma lógica **recursiva** para o tratamento da divisão da tarefa. Particularmente, dado o vetor original, o próprio algoritmo é chamado para suas partições esquerda e direita, para as quais retorna os extremos do intervalo ótimo e a soma maximizada. O **ponto base** dessa recursão, por sua vez, é o caso em que a partição contém

apenas um elemento, sendo sua solução a própria partição em si. O cálculo do subvetor que passa pelo ponto mediano, por sua vez, envolve percorrer as duas metades do vetor original linearmente até as extremidades, determinar até que ponto a soma é maximizada e juntar os resultados. Em detalhes, partindo do primeiro elemento à esquerda do "meio" do vetor, os demais são percorridos até que se alcance o primeiro e, para cada, a soma total até ali é verificada, armazenando-se a maior. O mesmo é feito do primeiro elemento à direita à extremidade desse mesmo sentido. O subvetor começa no elemento que maximiza a soma esquerda, termina no que maximiza a soma direita e apresenta soma total igual à soma dessas partições. Daí, a resposta para a instância geral do problema é a maior dentre as respostas para a partição esquerda, para a direita e para a central.

SUBVETORMAXIMO (comeco_do_vetor, fim_do_vetor, vetor)

SE comeco_do_vetor = fim_do_vetor **:**
RETORNAR (vetor[comeco_do_vetor], [comeco_do_vetor, fim_do_vetor])

CASO CONTRARIO :
meio_do_vetor = elemento mediano do vetor

(particao_esquerda, [comeco_e, fim_e]) = **SUBVETORMAXIMO** (comeco_do_vetor, meio_do_vetor, vetor)

(particao_direita, [comeco_d, fim_d]) = **SUBVETORMAXIMO** (meio_do_vetor + 1, fim_do_vetor, vetor)

PARA CADA elemento **DE** meio_do_vetor **A** comeco_do_vetor:
E = soma máxima partindo de meio_do_vetor
comeco_c = elemento que maximiza essa soma

PARA CADA elemento **DE** meio_do_vetor + 1 **A** fim_do_vetor
D = soma máxima partindo de meio_do_vetor + 1
fim_c = elemento que maximiza essa soma

(particao_central, [comeco_c , fim_c]) = (E + D, [comeco_c , fim_c])

RETORNAR max((particao_esquerda, [comeco_e, fim_e]), (particao_direita, [comeco_d, fim_d]), (particao_central, [comeco_c , fim_c]))

A razão para a escolha de uma solução recursiva baseada no paradigma de divisão e conquista se deve principalmente ao **ganho de eficiência** no algoritmo. Em particular, uma solução trivial para o problema consistiria em testar todos os possíveis subvetores e retornar aquele que maximiza a soma. O problema é que tal abordagem adotaria um **comportamento assintótico quadrático** com relação ao tamanho do vetor, o que pode vir a ser extremamente ineficiente para grandes entradas. Ao se dividir essa estrutura de dados pela metade a cada passo da recursão, verifica-se que são executados log (número de elementos) níveis recursivos, em cada qual os elementos do vetor são linearmente percorridos, originando uma complexidade **logarítmica * linear** no número de elementos da entrada.

3. Execução & Funcionamento

O programa começa sua execução declarando as variáveis que armazenarão os dados utilizados na solução de uma instância do problema, a saber, o **número de amigos no grupo** e o **número de shows no festival musical**. Aqui, pontua-se que várias instâncias podem ser

solucionadas em uma mesma execução, sendo necessário apenas fornecer essas informações corretamente formatadas pela entrada padrão. Tais dados serão utilizados para a inicialização de um objeto que representa o correspondente aplicativo de avaliações.

Em seguida, um **método** sobre o objeto referido é invocado para **coletar as informações** referentes às notas atribuídas por cada amigo a cada apresentação. Esses dados também são fornecidos via entrada padrão e **armazenados em uma matriz**, como já definido. Nesse mesmo procedimento, ainda, o **vetor contendo as notas gerais** para cada show também é devidamente inicializado e populado com informações calculadas a partir da estrutura de dados anterior.

Por fim, o **algoritmo para encontrar o intervalo de shows** que maximiza a satisfação do grupo é invocado sobre o objeto referente ao aplicativo de avaliações. Tal procedimento executa a sequência de passos tal qual como definida previamente e imprime na saída padrão o intervalo de shows propriamente dito, no formato **[índice_do_primeiro_show índice_do_último_show]**. O próprio objeto já está implementado de tal forma a realizar a limpeza das variáveis e a desalocação de memória ao fim da execução.

4. Considerações Finais

Instruções para compilação e para execução

1. Extraia o conteúdo do arquivo .zip disponibilizado.

2. Acesse o diretório **</tp2>** por meio do comando **<$ cd tp2>**.

3. Execute no terminal o comando **<$ make all>**. Esse comando deverá gerar um executável denominado **tp02**.

4. A execução do programa é realizada por meio do comando **<$ ./tp02 < input.txt>**, sendo **input.txt** o arquivo de texto contendo os dados de entrada.

5. A saída será retornada pela saída padrão de dados.

Referências Bibliográficas

Stack Exchange, Inc.. *Stack Overflow* [Online]. Disponível em: https://stackoverflow.com/ (Acessado em: 01/11/2022).

GeeksforGeeks. *Maximum Subarray Sum using Divide and Conquer algorithm* [Online]. Disponível em: https://www.geeksforgeeks.org/maximum-subarray-sum-using-divide-and-conquer-algorithm/ (Acessado em: 01/11/2022).