TP2 - Avaliação do Evento

Algoritmos 1

Departamento de Ciência da Computação - Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte - MG - Brazil
chbmleao@ufmg.br

1. **Identificação**

   Carlos Henrique Brito Malta Leão
   Matrícula: 2021039794

2. **Introdução**

   O trabalho a ser resolvido é um problema muito conhecido chamado de sublista contígua de soma máxima, em que, dado um arranjo de números, é preciso encontrar um subarranjo contíguo que contenha a maior soma que pode ser encontrado no arranjo completo. Por outro lado, geralmente esse problema requer apenas o valor dessa soma, já no contexto deste trabalho prático, precisamos encontrar um intervalo de elementos que maximize essa soma. Nesse sentido, existe um algoritmo muito conhecido para encontrar o valor de soma máxima, e, adicionando algumas funcionalidades ao código, também é possível encontrar o intervalo de soma máxima sem alterar a complexidade assintótica da função.

   Especificamente, no contexto deste trabalho prático, temos uma série de avaliações de m amigos, em que cada pessoa avalia um número n de shows. Dessa forma, precisamos definir qual intervalo de shows será o melhor aproveitado pelo conjunto de amigos, ou seja, qual intervalo de shows apresenta a maior avaliação somada. Para isso, primeiramente, é necessário formar um vetor com a avaliação geral de cada show, somando todas as avaliações deste. Em outras palavras, em um conjunto de duas amigos, se a pessoa A deu nota x para um show, e a pessoa B deu nota y para o mesmo show, esse show apresenta nota geral $x + y$, em que x e y são valores de -5 até 5.

   Por fim, ao gerar o vetor de avaliações gerais, é possível utilizar o algoritmo de sublista contígua de soma máxima para encontrar o melhor intervalo de shows para o conjunto de amigos. Esse algoritmo será melhor explicado e detalhado na seção seguinte. Além disso, existe mais um fator que deve ser considerado no algoritmo, em caso de empates na pontuação geral, show com pontuação geral zero, o conjunto de amigos sempre decide ir ao show.

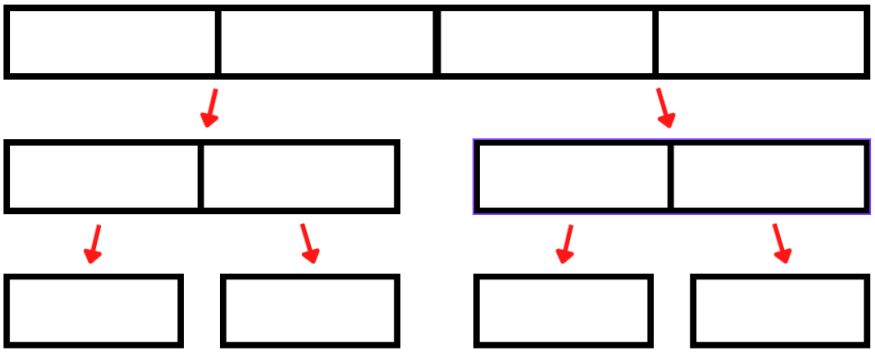
3. **Modelagem**

   Antes de utilizar o algoritmo supracitado, primeiro é preciso estruturar o vetor de avaliações gerais. Para isso, é preciso ler todas as m avaliações de n pessoas, em que m representa a quantidade de shows e n a quantidade de amigos. Nesse sentido, todas as avaliações para um determinado show x devem ser somadas ao elemento x do vetor de avaliações gerais. Ao final deste tratamento, teremos a avaliação geral de todos os shows, representa por um arranjo de pontos flutuantes de m elementos.

   Partindo para a parte do algoritmo principal do programa, precisamos de encontrar a sublista contígua de soma máxima do arranjo de avaliações gerais. Existem vários algoritmos para resolver esse problema, para esse trabalho, foi escolhido um algoritmo que utiliza uma estratégia muito famosa, conhecida como divisão e conquista. No contexto dessa estratégia, de forma geral, para resolver um

problema de tamanho n, resolva recursivamente m problemas de tamanho n/m e combine suas soluções para obter a resposta da versão completa do problema.

Primeiramente, precisamos dividir o problema em partes menores. Para isso, podemos dividir o vetor de pontuações gerais ao meio recursivamente, de forma que as últimas chamadas tratarão vetores de somente um elemento, que será o caso base da função. Segue uma figura que representa esse processo.



A partir da última divisão do vetor, é possível começar a resolver os subproblemas. Para isso, além de encontrar o valor da soma máxima e o seu intervalo no vetor, também será necessário descobrir o maior prefixo, o maior sufixo e a soma total do subvetor. Para conciliar melhor todas essas informações, será utilizada a classe Solution, em que essas quatro definições serão armazenadas cada uma em duas variáveis, sendo elas um ponto flutuante, que representa seu valor, e um par de inteiros que representa o intervalo no vetor. Essas informações serão melhor explicadas a seguir.

- **Soma:** armazena o valor da soma de todos os elementos do subvetor e o seu respectivo intervalo.
- **Prefixo:** armazena o valor da soma e o intervalo de um conjunto de elementos contíguos que contenham o elemento mais à esquerda do subvetor e que maximizem essa soma. Caso não seja possível formar um prefixo positivo com os elementos do subvetor, seu valor deve ser zero e o intervalo vazio, representado no código pelos valores -1 e -1.
- **Sufixo:** de forma contrária ao prefixo, armazena o valor da soma e o intervalo de um conjunto de elementos contíguos que contenham o elemento mais à direita do subvetor e que maximizem essa soma. Caso não seja possível formar um sufixo positivo com os elementos do subvetor, seu valor deve ser zero e o intervalo vazio, representado no código pelos valores -1 e -1.
- **Soma Máxima:** armazena o valor da soma e o intervalo de um conjunto de elementos contíguos que maximizem essa soma. Caso todos os elementos do subvetor sejam negativos, seu valor deve ser zero e o intervalo vazio, representado no código pelos valores -1 e -1.

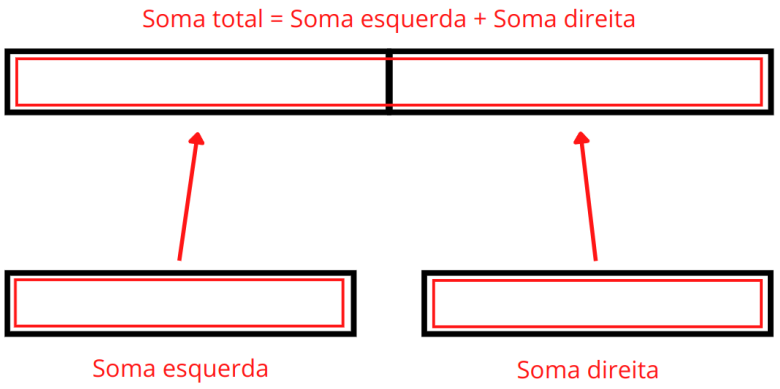
Essas informações devem ser determinadas para cada chamada recursiva do algoritmo, para assim conseguir determinar o valor e o intervalo de soma máxima do vetor completo. Nesse sentido, a função recursiva recebe como parâmetros o arranjo total e os índices do início e final do vetor. Segue uma explicação geral da função recursiva.

O **caso base** da função ocorre quando o subvetor apresenta apenas um elemento, ou seja, quando o início é igual ao final. Nesse caso, a **Soma** da solução é atribuída com o valor igual ao do único elemento do subvetor e, caso o elemento seja maior ou igual a zero, também é atribuído ao **Prefixo**, **Sufixo** e **Soma Máxima**, alterando também o intervalo destes para início até o final. Caso contrário, o valor 0 é atribuído a estes e o intervalo é vazio. Ao fim, a solução é retornada.

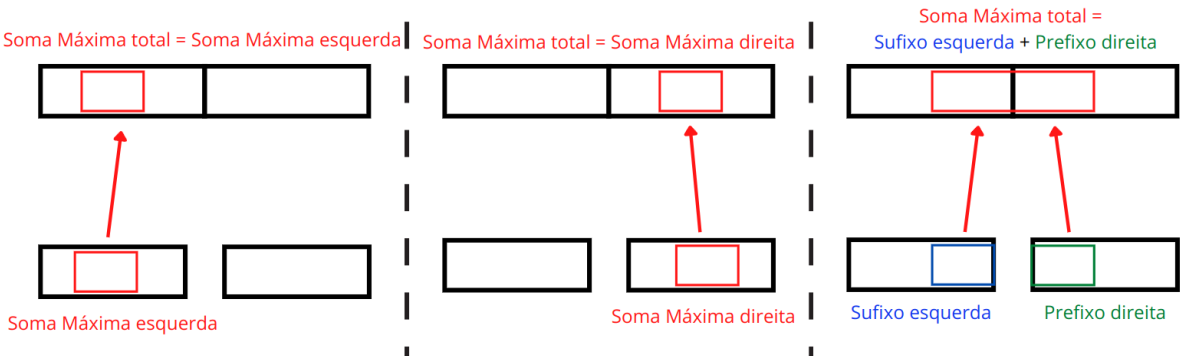
Já o caso contrário ao caso base é mais complexo, em que precisamos realizar um tratamento mais especializado para a determinação da **Soma Máxima**, do **Prefixo** e do **Sufixo**. Por outro lado, determinar a **Soma** segue uma lógica muito simples. Primeiramente, é preciso dividir o arranjo em dois, e determinar a solução para o subvetor da esquerda e direita. Essa solução apresenta a **Soma**, **Soma Máxima**, **Prefixo** e **Sufixo** do subvetor.

Antes de explicar como será feita a determinação de cada informação de um vetor, é preciso explicar como funciona a função de atualização de intervalos. Esse método recebe como parâmetros a identificação do intervalo que deve ser atualizado (**Soma**, **Soma Máxima**, **Prefixo** ou **Sufixo**), o novo índice de começo e fim do intervalo. Nesse sentido, a função atualizará o intervalo especificado pelo identificador com base nos índices indicados, porém, essa atualização ocorrerá, somente se ambos índices forem diferentes de -1, caso contrário, o intervalo não será atualizado, ou seja, continuará como um conjunto vazio.

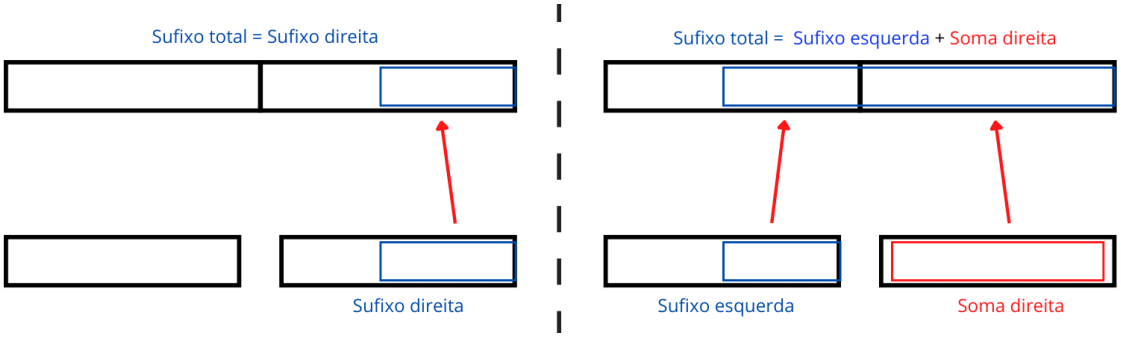
Dados o método de atualização de intervalo e as soluções da esquerda e direita, para determinar a **Soma** do arranjo original, basta somar a **Soma** da solução da esquerda com a da direita. Além disso, o intervalo completo será o começo do intervalo da esquerda e o final do intervalo da direita. Segue uma figura que demonstra esse processo de forma visual.



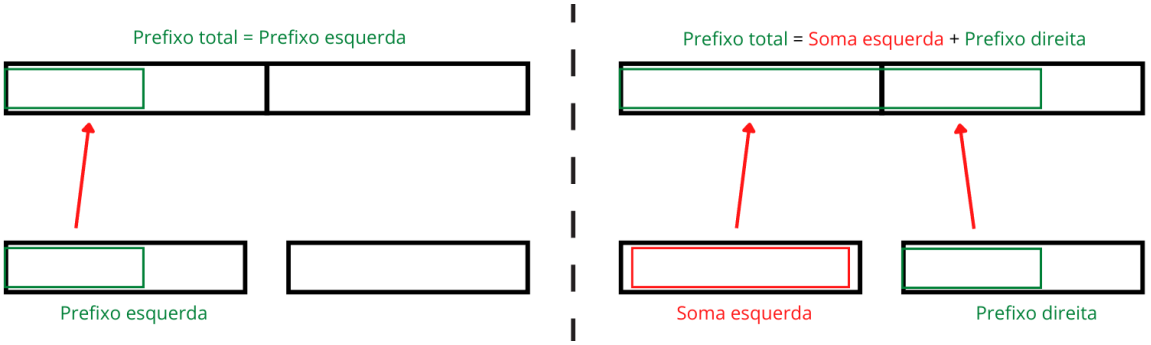
Para definir a **Soma Máxima** do arranjo original, existem três situações que podem ocorrer. As primeiras são simples, ocorrem quando a **Soma Máxima** do vetor completo é a mesma que ao subvetor da esquerda ou ao da direita, assumindo o mesmo valor e o mesmo intervalo. Por outro lado, também pode ocorrer que a **Soma Máxima** seja obtida a partir de uma junção do **Sufixo** da esquerda com o **Prefixo** da direita, recebendo o valor somado dos dois. Além disso, o intervalo será dado pelo começo do **Sufixo** da esquerda e final do **Prefixo** da direita. Nesse sentido, é necessário testar todas essas possibilidades para verificar qual apresenta a maior soma, esta, será a definição da **Soma Máxima** do vetor completo. Segue uma figura que demonstra as três possíveis situações.



Partindo agora para a definição do **Sufixo** do arranjo original, existem apenas duas situações que maximizam este. Pode ocorrer que o **Sufixo** do arranjo original seja o mesmo que o do subvetor da direita. Por outro lado, também pode ocorrer que o **Sufixo** do vetor original seja a **Soma** do subvetor da direita somada ao **Sufixo** do subvetor da esquerda, em que o intervalo será dado pelo início do **Sufixo** esquerdo até o fim do subvetor direito. Segue uma figura que demonstra as duas possíveis situações.



Por fim, falta apenas definir o **Prefixo** do arranjo original, em que este processo assemelha-se à definição do **Sufixo**, em que existem apenas duas situações a serem consideradas pelo algoritmo. Nesse sentido, pode ocorrer que o **Prefixo** do vetor original seja o próprio **Prefixo** do subvetor da esquerda. Além disso, também pode ocorrer do **Prefixo** do vetor completo ser a **Soma** do subvetor da esquerda somada ao **Prefixo** do subvetor da direita, em que o intervalo é dado pelo início do vetor da esquerda e o fim pelo fim do **Prefixo** da direita. Segue uma figura que demonstra as duas possíveis situações.



Assim, seguindo o processo supracitado, é possível encontrar o intervalo contíguo de soma máxima do vetor de avaliações gerais e, dessa forma, determinar o intervalo de shows consecutivos que mais agradou o grupo de amigos. Por fim, é

necessário realizar um tratamento adicional, já que, utilizando o algoritmo supracitado, se o arranjo apresentar apenas valores negativos, o subvetor de soma máxima retornado é um intervalo vazio. Porém, os amigos precisam ir em pelo menos um show do evento, dessa forma, se o intervalo retornado estiver vazio, é realizado uma iteração no vetor para encontrar o índice do maior elemento. Por fim, é impresso o intervalo do conjunto de apenas esse índice.

Por fim, o algoritmo descrito acima, utiliza em código, principalmente, a função maxSubarraySum, uma função recursiva que apresenta a complexidade de tempo igual a seguinte relação de recorrência: $T(n) = 2T(n/2) + O(n)$, que também pode ser representado pela complexidade assintótica de $O(n\,log\,n)$. Nesse aspecto, n representa a quantidade de shows apresentados no evento, como a quantidade de amigos é limitada por 50, esta não interfere na definição da complexidade assintótica do código. As funções para encontrar o maior valor e atualizar os intervalos apresentam tempo constante. Além disso, a leitura dos dados de entrada e o laço para descobrir o maior valor do vetor apresentam complexidade assintótica linear. Portanto, a complexidade assintótica de todo o algoritmo é $O(n\,log\,n)$, em que n representa o número de shows do evento. Essa complexidade deriva, principalmente, da utilização do paradigma de divisão e conquista para a resolução do problema.