TRABALHO PRÁTICO 3:
Show de Música

Pedro Araujo Pires

[1]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)

ppires@dcc.ufmg.br

Resumo. *Este relatório descreve duas soluções para o problema proposto para o trabalho prático 3 de AEDS 3. O problema consiste em, dado o número de músicas a serem tocadas em um show, a as variações de volume entre as músicas, achar a melhor sequência de variações de forma que a última música seja tocada no maior volume possível. Foram implementados um algoritmo força bruta e um algoritmo de programação dinâmica.*

1. INTRODUÇÃO

Neste trabalho foram implementados dois algoritmos (força bruta e programação dinâmica) para resolver o problema dos volumes em um show de música. O principal objetivo deste trabalho é comparar diferentes paradigmas de programação para resolver um determinado problema.

Em um show de música, o público fica mais empolgado se, entre duas músicas, houver alguma alteração no volume das músicas. O problema resolvido neste trabalho consiste em, dada uma sequência de alterações nos volumes de um show, achar uma determinada sequência de alterações de forma que a última música seja tocada no maior volume possível. Em nenhum momento o volume pode ficar menor do que 0, ou maior do que o volume limite.

Foram implementados dois algoritmos para resolver esse problema: força bruta e programação dinâmica. O algoritmo de força bruta testa exaustivamente todas as formas de se variar o volume, e armazena o melhor resultado. O algortmo de programação dinâmica calcula, para cada variação, qual é o maior volume possível de ser atingido na última música, e armazena esses valores. Ao final da execução, a resposta está na primeira linha da tabela.

O restante deste relatório é organizado da seguinte forma. A Seção 2 discute alguns temas relacionados ao trabalho. A Seção 3 descreve os algoritmos implmentados. A Seção 4 trata de detalhes específicos da implementação do trabalho. A Seção 5 contém a avaliação experimental dos algoritmos. A Seção 6 conclui o trabalho.

2. REFERÊNCIAS RELACIONADAS

Podemos dividir as referências associadas ao problema estudado e à solução proposta dentre os seguintes grupos:

- **Linguagem C:** Todo o trabalho foi implementado em linguagem de programação C. A linguagem C é uma linguagem de propósito geral amplamente utilizada no desenvolvimento de sistemas de alto desempenho, de tempo real, sistemas operacionais, compiladores, dentre outros. O compilador mais utilizado para a linguagem C é o GCC (desenvolvido pelo projeto GNU).

- **Projeto e análise de algoritmos:** Algoritmos são procedimentos computacionais capazes de resolver diversos problemas do mundo real. O problema de alterar o volume entre músicas durante um show, estudado neste trabalho, é um dos exemplos de como algoritmos estão presentes no nosso dia-a-dia. O estudo de algoritmos é essencial para o desenvolvimento de técnicas mais eficientes e eficazes para a solução desses problemas.

3. SOLUÇÃO PROPOSTA

A primeira solução proposta é a que utiliza um algoritmo força bruta. Essa solução varia o volume (soma ou subtrai) de todas as formas possíveis. Para isso, é utilizado um número binário para representar todas as formas possíveis de somar ou subtrair algum volume. Esse número binário possui n algarismos, onde n é o número de variações de volume durante o show, e ele é usado sa seguinte forma: se um determinado algarismo é 0, é efetuada uma subtração na variação de volume correspondente. Se for 1, o valor é somado. O número começa com o valor 0, e é incrementado de 1 unidade até $2^n - 1$.

3.1. Subestrutura ótima

O problema não possui subestrutura ótima.

3.2. Propriedade gulosa

O problema não possui propriedade gulosa.

3.3. Sobreposição de subproblemas

Há sobreposição de subproblemas. Sempre que, seguindo dois caminhos diferentes, se chegar no mesmo volume intermediário, há sobreposição de subproblemas. Foi desenvolvido, então, um algoritmo que utiliza programação dinâmica para resolver o problema. Esse algoritmo constrói uma tabela n x m, onde n é o número de variações de volume durante o show, e m é o volume limite + 1. Essa tabela armazena, para cada variação, qual é o maior volume possível de se tocar a última música, de acordo com o volume antes de se executar a variação. O processamento começa com a última variação, e calcula os resultados de trás para frente. Depois de construída a tabela, a solução do problema estará na primeira linha, na coluna correspondente ao volume inicial.

O preenchimento da tabela começa com a última variação. Para cada volume $(0..volumeLimite)$, a posição correspondente na tabela é preenchida com o volume inicial somado com a variação. Caso essa soma ultrapasse o volume limite, tenta-se subtrair a variação do volume inicial. Caso isso também não seja possível, coloca-se -1 na posição. Passa-se então para as outras linhas. Cada posição de i-ésima linha será preenchida com $max(tabela[i+1][vol + variação], tabela[i+1][vol - variação])$.

Como exemplo, considere o seguinte problema: 3 músicas a serem tocadas, no show, o volume limite é 5, o volume inicial é 3, e as variações de volume entre as músicas são 2 e 3. A tabela 1 mostra como a tabela fica depois do primeiro passo, e a tabela 2 mostra a tabela depois de terminada. A resposta para esse exemplo, para o volume inicial 1 é 0, e para o volume inicial 0 é 5.

	0	1	2	3	4	5
2						
3	3	4	5	0	1	2

Tabela 1. Tabela de programação dinâmica depois do primeiro passo

	0	1	2	3	4	5
2	5	0	3	4	5	0
3	3	4	5	0	1	2

Tabela 2. Tabela de programação dinâmica depois de totalmente preenchida

3.4. Estruturas de dados

3.4.1. Show:

Armazena informações sobre um show.

1: Show
número de músicas;
volume inicial;
volume limite;
variações de volumes entre as músicas;

3.5. Algoritmos

3.5.1. Força Bruta

Gera todas as soluções possíveis, para poder achar a maior. Como para cada intervalo entre músicas, a variação de volume pode ser somada ou subtraída, ao todo são 2^{n-1} formas diferentes de se variar o volume, onde n é o total de músicas tocadas no show. Para cada uma dessas formas, o resultado é calculado. Logo, o algoritmo força bruta possui uma complexidade de tempo $O(2^n)$. Para executar o cálculo, o algoritmo utiliza somente dois números inteiros, independente de quantas músicas o show possui. Logo, a complexidade de espaço é $O(1)$.

3.5.2. Programação Dinâmica

O algoritmo de programação dinâmica calcula, para cada variação de volume, qual é o maior volume possível de se tocar a última música. Para cada variação, os resultados são armazenados em uma tabela, e essa tabela é percorrida somente uma vez, durante seu preenchimento. Depois de preenchida, a resposta do problema estará na posição $[0][volumeInicial]$. Por percorrer a tabela somente uma vez, esse algoritmo possui uma complexidade de tempo $O((n-1)*(m+1)) = O(nm)$, onde n é o número de músicas, e m é o volume máximo permitido. Como a tabela para armazenar os valores possui $(n-1)*(m+1)$ posições, a complexidade de espaço deste algoritmo também é $O(nm)$.

4. IMPLEMENTAÇÃO

4.1. Código

4.1.1. Arquivos .c

- **io.c:** Define as funções relacionadas à parte de entrada e saída do programa.
- **benchmark.c:** Define as funções relacionadas à medição do tempo de execução.
- **show.c:** Define as funções relacionadas à strutura de dados Show.
- **fb.c:** Define as funções relacionadas ao algoritmo de força bruta.
- **pd.c:** Define as funções relacionadas ao algoritmo de programação dinâmica.

4.1.2. Arquivos .h

- **io.h:** Define os cabeçalhos das funções relacionadas à parte de entrada e saída do programa.
- **benchmark.h:** Define os cabeçalhos das funções relacionadas à medição do tempo de execução.
- **show.h:** Define a estrutura de dados Show, e os cabeçalhos das funções dessa estrutura.

4.2. Compilação

O programa deve ser compilado através do comando make. Esse comando irá compilar todos os arquivos, e gerar dois executáveis, $tp3\_fb$ e $tp3\_pd$. O primeiro resolve o problema utilizando o algoritmo de força bruta, e o segundo resolve o problema utilizando o algoritmo de programação dinâmica.

4.3. Execução

A execução dos programas recebe como parâmetro somente o nome do arquivo que contém os dados de entrada.

O comando para a execução do programa é da forma:

```
./tp3_fb <arquivo de entrada> [benchmark]
./tp3_pd <arquivo de entrada> [benchmark]
```

O parâmetro opcional $benchmark$, se passado, fará com que ao final da execução de cada instância do problema, seja impresso na tela o tempo de execução (em segundos) daquela instância.

4.3.1. Formato da entrada

O arquivo com os dados de entrada possui várias instâncias do problema. Cada instância ocupa duas linhas do arquivo, onde a primeira contém três inteiros (número de músicas, volme inicial e volume limite), e a segunda linha contém todas as variações de volume feitas durante o show, separadas por espaço.

```
4 5 10
5 3 7
5 8 20
15 2 9 10
```

4.3.2. Formato da saída

A saída do programa consiste em somente imprimir na tela o resultado da resolução do problema.

```
10
-1
```

5. AVALIAÇÃO EXPERIMENTAL

Para fazer a avaliação experimental dos algoritmos foram utilizadas duas entradas de teste. A primeira entrada foi utilizada para testar o algoritmo de força bruta, e ela contém várias instâncias do problema, onde o número de músicas é variado. Como esse parâmetro é o que determina a velocidade do algoritmo, essa é uma boa entrada para testar o algoritmo de força bruta. A Tabela 3 mostra o tempo de execução do algortimo força bruta para cada número de músicas na entrada. É possível verificar que a cada música adicionada ao show, o tempo de execução fica aproximadamente o dobro, confirmando a complexidade de $O(2^n)$ do algoritmo de força bruta.

# músicas	Tempo execução(seg.)
20	0,200524
21	0,387440
22	0,787293
23	1,596306
24	3,216643
25	6,530919
26	13,231659

Tabela 3. Tempo de simulação e do algoritmo força bruta

A segunda entrada utilizada para testes foi feita para testar o algoritmo de programação dinâmica. Nessa entrada o número de músicas e o volume máximo são variados, pois esses parâmetros são os que determinam o tempo de execução do algoritmo. A Tabela 4 mostra o tempo de execução do algortimo de programação dinâmica para alguns valores de volume máximo e de número de músicas no show. É possível verificar que, ao se dobrar um dos parâmetros, o tempo de execução também dobra. Também pode ser visto que, ao se dividir um dos parâmetros por 2, e dobrar o outro, o tempo de execução permanece muito parecido. Esse resultados confirmam a complexidade de $O(mn)$ feita para o algoritmo de programação dinâmica.

# músicas	Vol. máximo	Tempo execução(seg.)
26	512000	0,274157
51	512000	0,552829
26	1024000	0,547132
51	1024000	1,100637
26	2048000	1,100851
51	2048000	2,213717

Tabela 4. Tempo de execução do algoritmo de programação dinâmica

Todos os testes foram feitos em uma máquina com CPU Intel Core 2 T7400, com 2 processadores de 2.17GHz cada, rodando o sitema operacional Ubuntu Linux 10.04.

6. CONCLUSÃO

Neste trabalho foram desenvolvidos dois algoritmos para resolver o problema de variar o volume entre músicas, durante um show. O trabalho atingiu os objetivos propostos, que eram o estudo e implementação de diferentes paradigmas de programação para resolver um mesmo problema.

Foi possível observar que a utilização da programação dinâmica proporciona um ganho significativo na eficiência do algoritmo, comparativamente com o algoritmo força bruta. Infelizmente, essa técnica só pode ser utilizada quando há sobreposição de sub-problemas.