

# Atividade Prática 03

## Estruturas de Índices

---

Universidade Tecnológica Federal do Paraná (UTFPR), campus Apucarana  
Curso de Engenharia de Computação  
Disciplina de Estrutura de Dados 2 - EDCO4B  
prof. Dr. Luiz Fernando Carvalho  
Prof. Dr. Rafael Gomes Mantovani

---

### Instruções:

- Leia todas as instruções corretamente para poder desenvolver sua atividade/programa;
- Evite plágio (será verificado por meio de ferramentas automatizadas). Faça seu programa com os seus nomes de variáveis e lógica de solução. Plágios identificados anularão as atividades entregues de todos os envolvidos.
- Adicione comentários nos códigos explicando seu raciocínio e sua tomada de decisão. Porém, não exagere nos comentários, pois a própria estrutura do programa deve ser auto-explicativa.
- Salve sua atividade em um arquivo único, com todas as funções e procedimentos desenvolvidos. É esse **arquivo único** que deverá ser enviado ao professor.

## 1 Descrição da atividade

Depois de alguns semestres tendo aula com o professor M vocês perceberam que ele é uma pessoa bem normal. E como uma pessoa normal, muitas vezes seu humor varia ao longo dos dias: há dias mais desanimados e outros mais animados. Porém, uma estratégia para sempre tentar abstrair e ficar bem é ouvir música. É comum o professor M colocar um som de fundo nas suas aulas para que tanto ele, como os alunos, possam se distrair um pouco. E como todo viciado em música, professor M ficou curioso em descobrir as músicas mais tocadas em 2023 pelos usuários do Spotify. Felizmente, ele conseguiu encontrar essa informação com poucos cliques.

### 1.1 Dataset

Nesta atividade vocês irão manipular o dataset *Spotify 1.2M+ Songs*<sup>1</sup> que contém metadados de mais de 1 milhão músicas indexadas na plataforma. O conjunto de dados oferece

---

<sup>1</sup><https://www.kaggle.com/datasets/rodolfofigueroa/spotify-12m-songs>

uma variedade de recursos além do que normalmente está disponível em conjuntos de dados semelhantes. Ele fornece informações sobre os atributos, popularidade e presença de cada música em várias plataformas musicais. O conjunto de dados inclui informações como: nome da faixa, nome do(s) artista(s), data de lançamento, listas de reprodução e paradas do *Spotify*, e algumas estatísticas de *streaming*. A [Tabela 1](#) apresenta uma descrição detalhada de todos os campos existentes na descrição de cada música.

Tabela 1: Tabela de características disponíveis no arquivo “spotify-1M.csv”. A primeira coluna marca com um bullet (●) os campos que podem ter índices criados.

Índice	Nro	Campo	Descrição
	1	id	Identificador da música no <i>Spotify</i>
●	2	name	Nome da música
●	3	album	Album onde a música está contida
	4	album_id	Identificador do álbum no <i>Spotify</i>
●	5	artists	Nome do(s) artista(s) da música
	6	artists_ids	Lista de Identificador(es) do(s) artista(s) da música
●	7	track_number	número da faixa no correspondente álbum
●	8	disc_number	Número do disco do artista
●	9	explicit	Se a música contém conteúdo explícito ou não
	10	danceability_%	Porcentagem que indica quão adequada a música é para dançar
	11	energy_%	Nível de energia percebido da música
●	12	key	Clave (chave) da música
	13	loudness	Frequência (em decibéis) da maior parte da música
●	14	mode	Modo da música (maior ou menor)
	15	speechiness_%	Proporção de palavras faladas na música
	16	acousticness_%	Quantidade de som acústico na música
	17	instrumentalness_%	Quantidade de conteúdo instrumental na música
	18	liveness_%	Presença de elementos de performance ao vivo
	19	valence_%	Positividade do conteúdo musical da música
	20	tempo	Batidas por minuto, uma medida do andamento da música
	21	duration_ms	Duração da música, em milissegundos (ms)
	22	time_signature	assinatura temporal da música
●	23	year	Ano em que a música foi lançada
	24	release_date	Data completa que a música foi lançada (YYYY-MM-DD)

## 1.2 O que fazer?

Pensando em melhorar o seu processo de consulta às músicas, o professor M pediu para vocês desenvolverem **índices secundários** que permitam consultas dos valores existentes no arquivo. Por exemplo: retornar todas as músicas de um artista específico, retornar

todas as músicas lançadas em 2023, ou ainda retornar todas as músicas que possuem um índice de dançabilidade (*danceability*) acima de 10%. Essa é sua nova **missão**: desenvolver um programa com índices secundários que satisfaça as consultas de músicas do arquivo do professor M.

## 2 Entradas do programa

O programa receberá **três** arquivos texto como parâmetros: um arquivo de dados com as músicas a serem manipuladas (“**spotify-1M.csv**”), um arquivo de consulta, e um arquivo de saída. Abaixo, iremos detalhar cada um deles.

### 2.1 Arquivo de dados

O primeiro parâmetro é o arquivo de dados, o mesmo cuja estrutura é apresentada na [Tabela 1](#). O armazenamento das músicas é feito em um arquivo de extensão “.csv” (*comma separated values*). Cada registro possui um tamanho fixo, porém os campos são de tamanhos variáveis. Cada campo é separado por uma vírgula (‘,’) e os registros finalizados por uma quebra de linha. Há um registro de cabeçalho (*header*) que contém os nomes dos campos (atributos) dos registros. A [Figura 1](#) mostra uma imagem do arquivo de dados aberto em um editor de textos comum.

### 2.2 Arquivo de consulta (*query*)

Um arquivo texto contendo as informações necessárias para realizar a busca na estrutura de índices. Esse arquivo pode conter dois tipos de buscas:

- **consulta simples:** baseada em apenas um único tipo de índice. A [Figura 2](#) mostra um exemplo de consulta simples. Nesse caso, o arquivo possui duas linhas com conteúdo: na primeira linha o nome do campo que irá definir o índice secundário, e na segunda linha uma *string* com o conteúdo a ser buscado. No exemplo em questão, queremos criar um índice secundário com as informações dos nomes dos artistas e retornar todas aquelas que são da “Taylor Swift”.
- **consulta com operadores booleanos:** dois ou mais operadores booleanos concatenando diferentes índices para uma consulta mais robusta. Um exemplo de consulta booleana é apresentada na [Figura 4](#). Nesse caso, a primeira linha contém os nomes dos campos usados para criar vários índices secundários, separados por operadores booleanos. Os dois operadores válidos são: & que corresponde ao operador AND, e || que corresponde ao operador OU. Na segunda linha são passados os valores correspondentes de consulta em cada índice, separados por vírgulas. Na consulta especificada o usuário irá criar dois índices secundários distintos: um deles baseado no nome dos artistas (*artist\_name*), e o segundo baseado no ano de lançamento das músicas (*released\_year*). A consulta resultante irá levar em consideração a interseção dos resultados das buscas por Talor Swift no nome dos artistas, e 2020 no ano de lançamento. A

spotify-1M.csv

	id	name	album	album_id	artists	artist_ids	track_number	disc_number	explicit	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	duration_ms	time_signature	year	release_date
1	7lmeHLHBe4nmXzuXc0HDjk	Testify	The Battle Of Los Angeles	2eia0myWFgoHuttJytCgxX	['Rage Against The Machine']	['2d0hyoQ5ynDBnkvAbJKORj']	1	1	False	0.47	0.978	7	-5.399	1	0.0727	0.0261	1.09e-05	0.35600000000000004	0.503	117.906	210133	4.0	1999	1999-11-02
3	1wsRitfRRtWyEapl0q22o8	Guerrilla Radio	The Battle Of Los Angeles	2eia0myWFgoHuttJytCgxX	['Rage Against The Machine']	['2d0hyoQ5ynDBnkvAbJKORj']	2	1	True	0.599	0.9570000000000001	11	-5.763999999999999	1	0.188	0.0129	7.06e-05	0.155	0.489	103.68	206200	4.0	1999	1999-11-02
4	1hR0fIFK2qRG3f3RF70pb7	CaIm Like a Bomb	The Battle Of Los Angeles	2eia0myWFgoHuttJytCgxX	['Rage Against The Machine']	['2d0hyoQ5ynDBnkvAbJKORj']	3	1	False	0.315	0.97	7	-5.423999999999995	1	0.483	0.0234	2.03e-06	0.12	2.0.37	149.749	298893	4.0	1999	1999-11-02
5	2lbASgTS0D07MTuLAXlTW0	Mic Check	The Battle Of Los Angeles	2eia0myWFgoHuttJytCgxX	['Rage Against The Machine']	['2d0hyoQ5ynDBnkvAbJKORj']	4	1	True	0.44	0.9670000000000001	11	-5.83	0	0.237	0.163	3.64e-06	0.121	0.574	96.75200000000001	213640	4.0	1999	1999-11-02
6	1MQTmPY0Z6fcmQc56Hdo7T	Sleep Now In the Fire	The Battle Of Los Angeles	2eia0myWFgoHuttJytCgxX	['Rage Against The Machine']	['2d0hyoQ5ynDBnkvAbJKORj']	5	1	False	0.426	0.929	2	-6.729	1	0.0701	0.00162	0.105	0.078	9.0.539	127.059	205600	4.0	1999	1999-11-02
7	2LXPNLSMAauNJfnC58lSqY	Born of a Broken Man	The Battle Of Los Angeles	2eia0myWFgoHuttJytCgxX	['Rage Against The Machine']	['2d0hyoQ5ynDBnkvAbJKORj']	6	1	False	0.298	0.848	2	-5.947	1	0.0727	0.0538	0.0015199999999999999	9.0.201	0.19399999999999998	148.282	280960	4.0	1999	1999-11-02
8	3moekHk8eIajvUEzVocXukf	Born As Ghosts	The Battle Of Los Angeles	2eia0myWFgoHuttJytCgxX	['Rage Against The Machine']	['2d0hyoQ5ynDBnkvAbJKORj']	7	1	False	0.41700000000000004	0.976	9	-6.032	1	0.175	0.0004269999999999999	97.0.000134	0.107	0.483	90.395	202040	4.0	1999	1999-11-02
9	4llunZfVxv3NvUzXVB3VVL	Maria	The Battle Of Los Angeles	2eia0myWFgoHuttJytCgxX	['Rage Against The Machine']	['2d0hyoQ5ynDBnkvAbJKORj']	8	1	False	0.27699999999999997	0.873	11	-6.57100000000000015	0	0.0883	0.00694000000000001	5.4000000000000001e-05	0.188	0.618	172.84799999999996	228093	4.0	1999	1999-11-02
10	21Mq0NzFoVRvOmLTOnJjng	Voice of the Voiceless	The Battle Of Los Angeles	2eia0myWFgoHuttJytCgxX	['Rage Against The Machine']	['2d0hyoQ5ynDBnkvAbJKORj']	9	1	False	0.441	0.882	7	-7.362999999999999	1	0.044000000000000004	0.0195	0.006840000000000001	0.15	0.418	83.371000000000002	151573	4.0	1999	1999-11-02
11	6s2FgJbnnMwFTpwJZzv6z	New Millennium Homes	The Battle Of Los Angeles	2eia0myWFgoHuttJytCgxX	['Rage Against The Machine']	['2d0hyoQ5ynDBnkvAbJKORj']	10	1	False	0.44799999999999995	0.861	9	-6.12	1	0.0676	0.003060000000000002	0.0.0.0.0.0987	0.7609999999999999	92.777	224933	4.0	1999	1999-11-02	
12	7o2Razxn133Jrgz4PpMYNa	Ashes In the Fall	The Battle Of Los Angeles	2eia0myWFgoHuttJytCgxX	['Rage Against The Machine']	['2d0hyoQ5ynDBnkvAbJKORj']	11	1	True	0.456	0.7040000000000001	7	-6.687	1	0.0982	0.0052	4.12e-0							

Figura 1: Dados de entrada do arquivo “spotify-1M.csv”

ordem de procedência dos operadores é da esquerda para a direita, conforme forem definidos na busca.

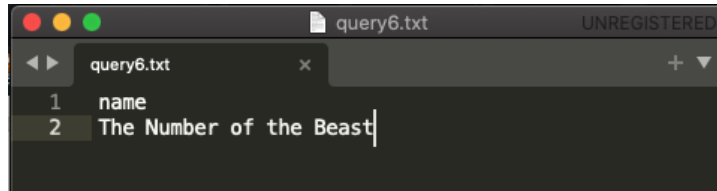
Os campos passíveis de criação de índices secundários estão identificados na [Tabela 1](#). Caso qualquer arquivo de consulta possua uma *string* indicando a criação de tipo de índice de campos inválido, erros devem ser capturados e indicados no arquivo de saída.

## 2.3 Arquivo de saída

Um arquivo texto contendo a busca realizada pelo programa após criar o correspondente índice secundário. A [Figura 3](#) mostra um exemplo do arquivo de saída onde são retornadas todas as músicas cujo nome da canção é “*The Number of the Beast*” contidas no arquivo de dados (base de dados) para o arquivo de busca “query1.txt” da [Figura 2](#). No arquivo de saída aparecem apenas 2 registros retornados pela busca.

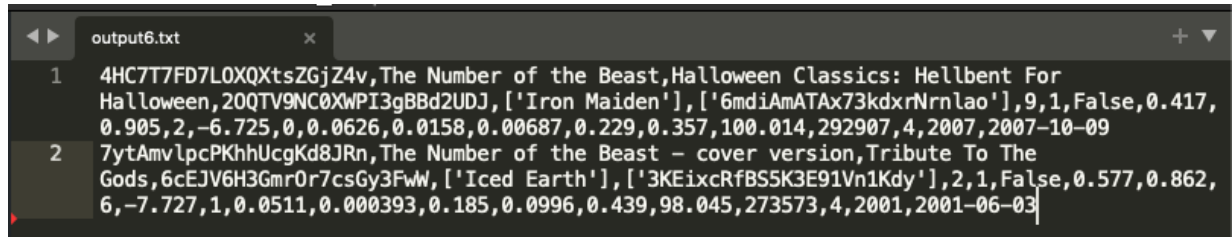
Já a [Figura 5](#) mostra a saída quando o arquivo de busca “query2.txt” da [Figura 4](#) é executado. Nesse caso são procuradas todas as músicas que sejam do artista = “*Rage Against the Machine*”, que foram lançadas no ano de 1992, e cujo título da canção é “*Bombtrack*”. O resultado da busca retorna um total de 4 registros.

Aqui é importante salientar também que algumas buscas podem simplesmente não retornar nenhum valor. Nesse caso, basta imprimir no arquivo de saída a seguinte mensagem:



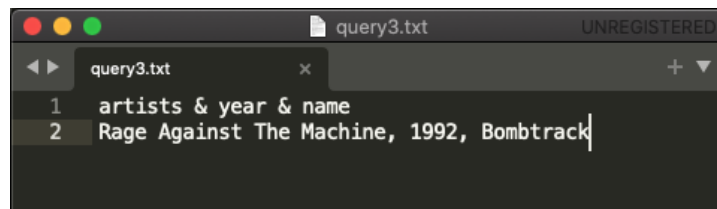
```
query6.txt
1 name
2 The Number of the Beast
```

Figura 2: Exemplo de arquivo de consulta simples.



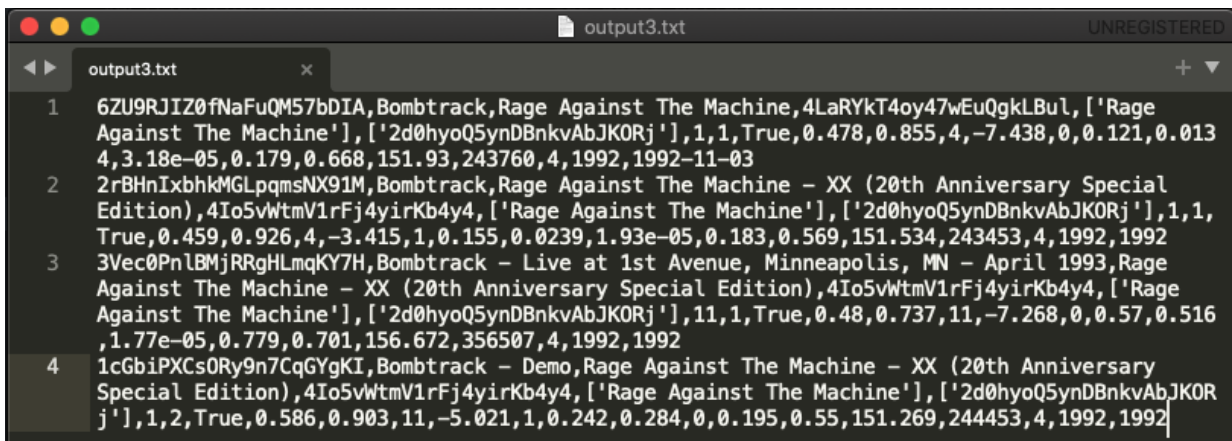
```
output6.txt
1 4HC7T7FD7L0XQXtsZGjZ4v,The Number of the Beast,Halloween Classics: Hellbent For
Halloween,20QTV9NC0XWPI3gBBd2UDJ,['Iron Maiden'],['6mdiAmATAx73kdxrNrnla0'],9,1,False,0.417,
0.905,2,-6.725,0,0.0626,0.0158,0.00687,0.229,0.357,100.014,292907,4,2007,2007-10-09
2 7ytAmvlpCpKhhUcgKd8JRn,The Number of the Beast - cover version,Tribute To The
Gods,6cEJV6H3Gmr0r7csGy3Fw,['Iced Earth'],['3KEixcRfBS5K3E91Vn1Kdy'],2,1,False,0.577,0.862,
6,-7.727,1,0.0511,0.000393,0.185,0.0996,0.439,98.045,273573,4,2001,2001-06-03
```

Figura 3: Saída correspondente gerada pelo programa para a *query* apresentada na [Figura 2](#).



```
query3.txt
1 artists & year & name
2 Rage Against The Machine, 1992, Bombtrack
```

Figura 4: Exemplo de arquivo de consulta com operador booleano.



```
output3.txt
1 6ZU9RJIZ0fNaFuQM57bDIA,Bombtrack,Rage Against The Machine,4LaRYkT4oy47wEuQgkLBul,['Rage
Against The Machine'],['2d0hyoQ5ynDBnkvAbJKORj'],1,1,True,0.478,0.855,4,-7.438,0,0.121,0.013
4,3.18e-05,0.179,0.668,151.93,243760,4,1992,1992-11-03
2 2rBHnIxbhkMGLpqmsNX91M,Bombtrack,Rage Against The Machine - XX (20th Anniversary Special
Edition),4Io5vWtmV1rFj4yirKb4y4,['Rage Against The Machine'],['2d0hyoQ5ynDBnkvAbJKORj'],1,1,
True,0.459,0.926,4,-3.415,1,0.155,0.0239,1.93e-05,0.183,0.569,151.534,243453,4,1992,1992
3 3Vec0PnlBMjRRgHLMqKY7H,Bombtrack - Live at 1st Avenue, Minneapolis, MN - April 1993,Rage
Against The Machine - XX (20th Anniversary Special Edition),4Io5vWtmV1rFj4yirKb4y4,['Rage
Against The Machine'],['2d0hyoQ5ynDBnkvAbJKORj'],11,1,True,0.48,0.737,11,-7.268,0,0.57,0.516
,1.77e-05,0.779,0.701,156.672,356507,4,1992,1992
4 1cGbiPXCs0Ry9n7CqGYgKI,Bombtrack - Demo,Rage Against The Machine - XX (20th Anniversary
Special Edition),4Io5vWtmV1rFj4yirKb4y4,['Rage Against The Machine'],['2d0hyoQ5ynDBnkvAbJKOR
j'],1,2,True,0.586,0.903,11,-5.021,1,0.242,0.284,0,0.195,0.55,151.269,244453,4,1992,1992
```

Figura 5: Saída correspondente gerada pelo programa para a *query* apresentada na [Figura 4](#).

'Nenhum resultado foi encontrado!'. Além disso, em caso de consultas inválidas, arquivos corrompidos ou inválidos, o arquivos de saída devem conter uma mensagem indicando o erro capturado.

### 3 Rodando o programa

Para rodar o programa por linha de comando você deve manipular os argumentos em *Python* por meio do parâmetro **sys.argv**. Para executar o programa por linha de comando, deve-se obedecer o seguinte padrão:

```
[nome do programa] [arquivo de dados] [arquivo de entrada] [arquivo de saída]
```

Exemplo de execução de um programa chamado `indice.py`:

```
python3 indice.py spotify-1M.csv query01.txt saida01.txt  
python3 indice.py spotify-1M.csv query02.txt saida02.txt
```

### 4 Orientações gerais

Além da funcionamento da estrutura de índice secundário desejada, implementar também o controle de erros para lidar com exceções que possam ocorrer, como por exemplo:

- problemas nas aberturas dos arquivos de entrada e saída;
- arquivos de entrada vazio (sem informação);
- arquivos de entrada fora do padrão esperado (opções inválidas para consulta);
- número de argumentos inválido na linha de comando (mais ou menos valores);
- etc.

Opcionalmente, para acompanhamento do desenvolvimento, pode-se criar um repositório individual no `github`.

### 5 Critérios de correção

A nota na atividade será contabilizada levando-se em consideração alguns critérios:

1. pontualidade na entrega;
2. não existir plágio;
3. completude da implementação (tudo foi feito);
4. o código executa;
5. uso de `sys.arg` para controle dos arquivos de teste;
6. implementar a leitura dos dados de entrada via arquivo texto (musicas);
7. implementação correta das estruturas necessárias (campos, registros e sua manipulação, ordenação das chaves, estrutura de índice);
8. legibilidade do código (identação, comentários nos blocos mais críticos);
9. implementação dos controles de erros (arquivos de entrada inválidos, e erros no programa principal);
10. controle de memória: chamar o destrutor e desalocar a memória de tudo se usar estruturas dinâmicas, fechar os arquivos, etc;
11. executar corretamente os casos de teste.

Em cada um desses critérios, haverá uma nota intermediária valorada por meio de conceitos:

- **Sim** - se a implementação entregue cumprir o que se esperava daquele critério;
- **Parcial** - se satisfizer parcialmente o tópico;
- e **Não** se o critério não foi atendido.

## 6 Padrão de nomenclatura

Ao elaborar seu programa, crie um único arquivo fonte (.py) seguindo o padrão de nome especificado:

ED2-AT03-IndiceSecundario-<NOME>.py

Exemplo:

ED2-AT03-IndiceSecundario-Maurichan.py

A entrega da atividade será via Moodle: o link será disponibilizado na página da disciplina.



## 7 Links úteis

- Arquivos em Python:

- <https://www.geeksforgeeks.org/reading-writing-text-files-python/>
- [https://www.w3schools.com/python/python\\_file\\_open.asp](https://www.w3schools.com/python/python_file_open.asp)
- <https://www.pythontutorial.net/python-basics/python-read-text-file/>

- Argumentos de Linha de comando no Python:

- [https://www.tutorialspoint.com/python3/python\\_command\\_line\\_arguments.htm](https://www.tutorialspoint.com/python3/python_command_line_arguments.htm)
- <https://realpython.com/python-command-line-arguments/>
- <http://devfuria.com.br/python/sys-argv/>

- *Spotify datasets*:

- <https://www.kaggle.com/datasets/nelgiriyeewithana/top-spotify-songs-2023>
- <https://www.kaggle.com/code/nelgiriyeewithana/an-introduction-to-top-spotify-songs-2023>
- <https://www.kaggle.com/datasets/rodolfofigueroa/spotify-12m-songs>

## Referências

- [1] Michael J. Folk; Bill Zoellick; Greg Riccardi. File Structures, 3rd edition, Addison-Wesley, 1997.
- [2] Thomas H. Cormen,; Ronald Rivest; Charles E. Leiserson; Clifford Stein. Algoritmos - Teoria e Prática - 3ª Ed. Elsevier - Campus, 2012.
- [3] Nivio Ziviani. Projeto de algoritmos com implementações: em Pascal e C. Pioneira, 1999.
- [4] Adam Drozdek. Estrutura De Dados e Algoritmos em C++. Cengage, 2010.