Mestrado em Engenharia Informática
Sistemas de Gestão de Base de Dados

Árvores balanceadas n-árias

Trabalho prático 2 – Variante 2

Neste trabalho pretende-se árvores n-áreas balanceadas (*B⁺-Tree*), onde os dados estão armazenados numa lista duplamente ligada e as (entradas das) folhas da árvore contêm ponteiros para os respectivos registos da lista.

Com a concretização deste trabalho os alunos devem: 1) ser capazes de pesquisar dados via árvore; 2) inserir/eliminar entradas; e 3) compreender o uso de índices arborescentes.

Trabalho preparatório

Dados sobre o disco (sectores e blocos)

- Sectores 4K:
  http://www.seagate.com/tech-insights/advanced-format-4k-sector-hard-drives-master-ti/

- Sector lógico vs. Sector Físico:
  https://docs.microsoft.com/en-us/windows/compatibility/advanced-format-disk-compatibility-update

- Obter informação sobre o disco (get the sector size info within an elevated command prompt)
  fsutil fsinfo ntfsinfo <drive letter>
  msinfo32 ("System Summary->Components->Storage->Disks")

- Cylinder-head-sector (CHS)
  https://en.wikipedia.org/wiki/Cylinder-head-sector

- Disk sector:
  https://en.wikipedia.org/wiki/Disk_sector

- Physical record:
  https://en.wikipedia.org/wiki/Block_(data_storage)

Consultas bibliográficas (disco e B⁺-Tree)

- [1] Feliz Gouveia, "Bases de Dados – Fundamentos e Aplicações", 2ª Edição, FCA, 2021, ISBN: 978-972-722-901-7.
  (Pág. 145-174)

- [2] R. Ramakrishnan & J. Gehrke, "Database Management Systems", 3rd Edition, McGraw-Hill, 2003. ISBN: 0-07-246563-8.
  (Pag. 306-308; 338-358)

Importação de dados (.csv)

- *Libcsv is a small, simple and fast CSV library written in pure ANSI C89 that can read and write CSV data*.
  https://sourceforge.net/projects/libcsv/

- Google it: "*c read csv file into struct*"

Implementação de B^+-Tree

- B^+-tree.
  https://www.programiz.com/dsa/b-plus-tree

- Algoritm in C – B+ Tree
  https://setscholars.net/algorithm-in-c-b-tree-2/

- Introduction of B+ Tree
  [3] https://www.geeksforgeeks.org/introduction-of-b-tree/
  [4] https://www.geeksforgeeks.org/insertion-in-a-b-tree/

- Google it: "*B^+-tree implementation in C*"

Dataset

https://transtats.bts.gov/PREZIP/On_Time_Reporting_Carrier_On_Time_Performance_1987_present_2021_7.zip

Tarefas

O grupo de trabalho deve escolher uma linguagem de programação com que se sinta confortável (recomenda-se C/C++, devido à flexibilidade no uso de ponteiros).

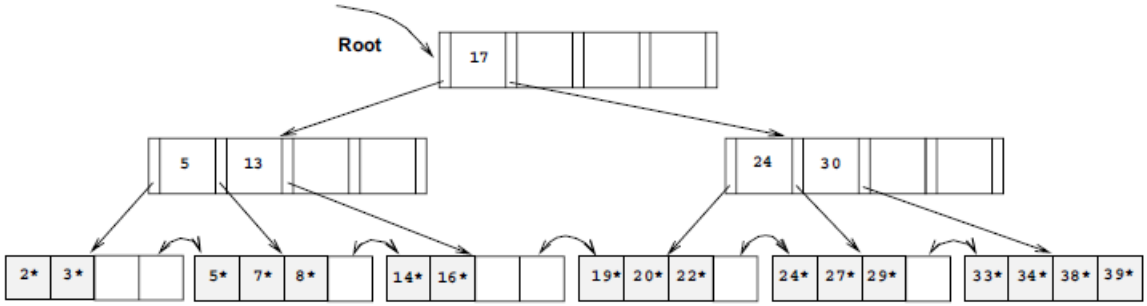
Podem usar excertos de código da internet, directamente ou adaptados. A aplicação deve permitir.

1. Ler o ficheiro .csv para uma lista duplamente ligada (designada por *DLdata*). Cada nó da lista, para além dos dois ponteiros (*Prev* e *Next*), deve conter um registo com os primeiros 30 campos de cada linha.

   Nota: pode ser usada qualquer estratégia para carregar os dados: 1) usar uma biblioteca para o efeito; 2) escrever uma função; 3) carregar os dados para uma tabela da base de dados e depois lê-los; ou 4) …

2. Pretende-se construir uma B^+-*tree* (*BPTairport*), usando como chave de pesquisa o campo *DestAirportID*. Suponha que cada nó ocupa 4 KB (4096 byte). O número de entradas por nó deve ser calculado tendo em conta o tamanho da chave de pesquisa e o tamanho dos ponteiros.

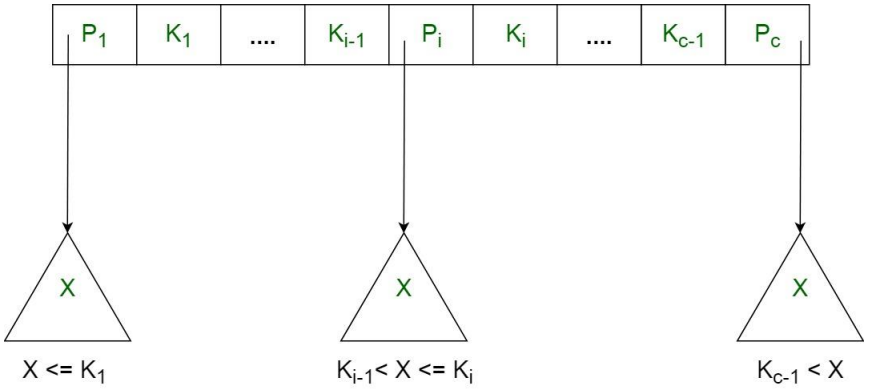
   A imagem seguinte ilustra um exemplo da estrutura B^+-*tree*, considerando 4 entradas por nó.



(Extraído de [2], pág. 351)

   Os nós não-folha têm a seguinte estrutura:

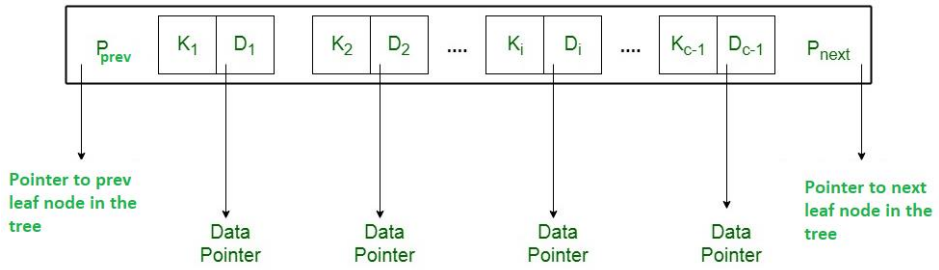
   $<P_1, K_1, P_2, K_2, \ldots, P_{c-1}, K_{c-1}, P_c>$, onde P_i é um ponteiro para uma subárvore e K_i é uma chave de pesquisa (*DestAirportID*).



(Extraído de [4], 2021-10-21)

Os nós folha têm a seguinte estrutura:

$<P_{prev}, <K_1, D_1>, <K_2, D_2>, ..., <K_{c-1}, D_{c-1}>, P_{next}>$, onde onde D_i é um ponteiro para um nó da lista *DLdata* e K_i é uma chave de pesquisa (*DestAirportID*). P_{prev} e P_{next} permitem implementar uma lista duplamente ligada.



(Adaptado de [4], 2021-10-21)

```
// Bplus Tree Node
typedef struct BPnode {
  void **pointers;
  int *keys;
  //struct BPnode *parent;
  bool is_leaf;
  int num_keys;
  struct BPnode *Pprev;
  struct BPnode *Pnext;
} BPnode
```

Nota: `void **pointers`, com o *Cast* apropriado tanto pode ser usado para ponteiros para a subárvore como para ponteiros para os nós da lista *DLdata*.

3. A árvore *BPTairport* deve ser construída para "indexar" a lista *DLdata*. Ou seja, a aplicação deve percorrer a lista *DLdata* e criar as correspondentes entradas na árvore.

4. Construída a árvore, a aplicação deve permitir aceitar um aeroporto de destino e apresentar os dados dos voos para esse aeroporto.

5. Análise de resultados

   Comparar os tempos consumidos no varrimento sequencial da lista *DLdata* e da pesquisa usando a árvore *BPTairport*. Fazer 10 testes com diferentes aeroportos de destino. Calcular a média e o desvio padrão.

Relatório

Elaborar um relatório com a descrição pormenorizada do trabalho realizado.

Elementos a entregar

Via Moodle, para a pasta apropriada.

Estrutura do Relatório

O relatório deve conter, pelo menos, os seguintes capítulos:

Capa

Identificação dos elementos do grupo.

1. Introdução

   Apresentação do trabalho desenvolvido e introdução genérica às ferramentas utilizadas.

2. Estruturas de dados

   Construção da lista *DLdata* a partir do ficheiro .csv.

   Construção da árvore *BPTairport*, incluindo o cálculo do número de entradas.

3. Exploração da aplicação

   Comparação de resultados.

4. Conclusões

   Reflexão crítica sobre os resultados.