Revisão P2 Pesquisa e Ordenação de Dados Giancarlo

Erickson G. Müller

November 28, 2024

Conteúdos

1. Busca Linear e Binária
2. Tabela Hash
3. Árvore B
4. Árvore B+
5. Compressão de Dados

1 Busca Linear e Binária

A **busca linear** acontece em um vetor ou lista encadeada.

- Melhor caso: $O(1)$
- Pior caso: $O(n)$
- Caso médio: $O(\frac{n+1}{2})$

A **busca binária** só pode ser implementada em vetores ordenados. Que ocorre através de **divisão e conquista**. Partindo do meio do vetor, divide-o até encontrar ou não o elemento.

- Melhor caso: $O(1)$
- Pior caso: $O(\log n)$
- Caso médio: $O(\log n)$

2 Tabela Hash

Em tabelas hash, a complexidade de busca é sempre $O(1)$. Também chamada de tabela de dispersão ou tabela de espalhamento, a tabela hash armazena uma ou mais chaves(e seus valores associados) em um vetor. Os elementos ficam dispostos de forma **não ordenada**.
Espalha-se os dados em uma grande tabela usando uma função cujo objetivo é evitar que esses dados caiam no mesmo índice da tabela (colisão). A função *hashing* transforma cada chave em um inteiro equivalente a um dos índices da tabela hash.
Em caso de colisões, pode-se criar uma lista encadeada em cada índice da tabela:

2.1 Tratamento de Colisões

- **Endereçamento aberto**: Em caso de colisão, a chave é adicionada em outra posição da tabela, facilitando a busca. Em contrapartida, há a necessidade de implementar algoritmo auxiliar para calcular a posição na tabela, a seguir alguns algoritmos:

  1. Sondagem linear: será adicionado na próxima posição livre da tabela (Ao K da função hashing será adicionado um j).
  2. Sondagem quadrática: para evitar a formação de cluster primários, similar à sondagem linear, contudo, ao K da função hashing será adicionado um j^2.
  3. Hashing duplo: utiliza uma segunda função hash para gerar um resultado diferente.

Inserção em endereçamento aberto

- Melhor caso: $O(1)$
- Pior caso: $O(n)$, quando todas as chaves são mapeadas para posições ocupadas.

Busca em endereçamento aberto

- Melhor caso: $O(1)$
- Pior caso: $O(m)$, pois terá que testar todas as posições até encontrar a chave.

Remoção em endereçamento aberto

- Melhor caso: $O(1)$
- Pior caso: $O(m)$, pois pode ser necessário testar todas as posições para encontrar a chave.

- **Encadeamento separado**: Hashing aberto, cada posição da tabela aponta para o início de uma lista encadeada. Necessita de memória adicional àquela que foi alocada para a tabela. **Inserção em encadeamento separado**

  - Melhor caso: $O(1)$
  - Pior caso: $O(1)$, se a lista não for ordenada.

  Busca em encadeamento separado

  - Melhor caso: $O(1)$
  - Pior caso: $O(n)$, quando todas as chaves estão na mesma posição do vetor e a chave procurada estiver no final da lista.

  Remoção em encadeamento separado

  - Melhor caso: $O(1)$
  - Pior caso: $O(n)$, quando todas as chaves estão na mesma posição do vetor e a chave procurada estiver no final da lista.

  Podemos reparar que as operações com encadeamento separados são menos complexas no pior caso que as operações com endereçamento aberto. A pior desvantagem é memória adicional que é alocada dinamicamente nas listas encadeadas, em vez de aproveitar ao máximo a memória destinada à tabela.

3 Árvore Binária