Introdução

No presente trabalho a base de dados short abstracts do wikipedia pt é indexada, utilizando uma tabela hash e uma lista duplamente ligada em segundo nível.

Metodologia

O arquivo database.ttl é lido, cada linha é tomada como uma entrada de onde o doc_id variável que identifica o documento é extraido, caracteres especiais são filtrados e transformados em similares presentes na tabela ASC, então cada palavra vai para uma lista que guarda o número de repetições da palavra naquela entrada, posteriormente cada palavra é buscada na tabela hash se a entrada já existir as ocorrências são adicionadas e se não existir são inseridas na tabela hash.

Um vetor de ponteiros serve como tabela hash, os índices reais são mapeados por uma função hash que pode ser selecionada durante criação da instância da tabela. Cada lista apontada pela tabela hash contém os números de palavras da chave encontradas para cada arquivo indexado, arquivos cujo número de palavras encontradas é zero não entram na lista.

Compilação

Para compilar os fontes deve-se utilizar os parâmetros -std=c99, referente a utilização do padrão C99 e o parâmetro -w, que elimina mensagens de warning. Os fontes não dependem de nenhuma biblioteca de terceiros.

Experimentos

A solução hash tem pior caso O(n) e aninhada uma busca em lista duplamente encadeada O(n) , sendo assim o pior caso é O(n^2).

Para viabilizar a experimentação é feita com 10000 linhas da base de dados e os tempos de execução para indexação com cada uma das funções hash é comparado.

Resultados

A função hash de Robert Sedgwicks (Algorithms in C book)

Levou 2 minutos e 0.085 segundos para indexar 10000 entradas.

A função hash de Justin Sobel

Levou 2 minutos e 0.286 segundos para indexar 10000 entradas.

A função hash de Brian Kernighan and Dennis Ritchie's (The C Programming Language)

Levou 1 minuto e 58.778 segundos para indexar 10000 entradas.

A função hash de Professor Daniel J. Bernstein

Levou 2 minutos e 1.062 segundos para indexar 10000 entradas.

A função hash ELF

Levou 2 minutos e 2.929 segundos para indexar 10000 entradas.

Conclusão

A função hash de Brian Kernighan and Dennis Ritchie's apresenta o menor tempo, sendo uma boa candidata para solução do problema em questão. O consumo de memória é similar para todas as funções e proporcional ao tamanho da entrada em ordem O(n^2). Pode-se observar o decaimento do desempenho enquanto a tabela hash cresce, já que as buscas subsequentes têm de ser feitas em estruturas maiores.

Trabalhos Futuros

Uma melhor solução pode apresentada utilizando paralelismo, dividindo a base de dados em bases menores e distribuindo a carga de indexação em várias unidades de processamento, clusters de processamento são, por fim, uma boa alternativa.

Referências Bibliográficas

- http://www.partow.net/programming/hashfunctions/#AvailableHashFunctions, Acessado em 31/10/2015
- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, *Algoritmos Teoria e PráticaI*, 2ª ed.
- Nivio Ziviane, *Projeto de Algoritmos com implementações em Java e C++*
- Robert Sedgewick and Kevin Wayne, *Algorithms*, 4th Edition
- Brian Kernighan and Dennis Ritchie's, *The C Programming Language*, 2ª ed