Felipe Machado Cordeiro

Universidade Federal de Minas Gerais

Matemática Computacional

Introdução

O problema apresentado consiste em criar um índice invertido a partir de vários arquivos de textos usando somente uma certa quantidade de memória disponibilizada.

Um índice invertido é um arquivo de texto que contém todas palavras dos arquivos originais e apresenta de qual documento a palavra veio, a frequência e a posição que ela aparece em seu arquivo de origem.

Para esse problema foi usado os conceitos de ordenação externa por meio de Seleção por substituição para ordenar e depois criar o índice.

Solução

Usou-se a ordenação externa para ordenar todas palavras em um arquivo, considerando a palavra, documento de origem e posição para a ordenação. Depois de ordenar os registros, calculou-se a frequência de cada palavra em seu arquivo e a escreveu no índice.

Na ordenação externa, primeiro lia-se o máximo de palavras que cabem na memória disponibilizada. Depois cria-se a propriedade de heap nesse vetor deixando o menor registro no topo. Escreve o topo em um arquivo, e lê o próximo registro das conversas. Se o registro lido for menor que o topo da fila, marca-se esse registro e o insere no topo do heap, ajustando-se depois para manter a propriedade. Arquivos marcados são sempre maiores que os não marcados. Quando um arquivo marca chega topo do heap, deve-se abrir um novo arquivo de saída.

Quando todos registros dos arquivos de conversas forem lidos, estes são fundidos em um arquivo que contém todos registros ordenados.

Para realizar isto, abre-se o máximo de arquivos possíveis, que é limitado pela quantidade de memória disponibilizada para os registros e pelo máximo de arquivos possíveis de abrir simultaneamente, e lê o primeiro registro de cada arquivo. Depois criar a propriedade de heap com esses arquivos. Retira o topo e lê o próximo registro do arquivo que o registro saiu. Caso não tenha mais registros a serem lido no arquivo simplesmente retira-se o registro do heap, assim esse arquivo se torna inativo. Repete-se isso até que todos arquivos se encontrem em um arquivo.

Com esse arquivo com todos registros ordenados, calcula-se a frequência de cada palavra e escreve o registro em no arquivo de índice com a frequência.

Para se calcular a frequência, um registro é lido, se o próximo registro e do mesmo documento e possui a mesma palavra, aumenta frequência e faz o mesmo. Caso contrário escreve os registros iguais no índice e faz o mesmo para o próximo registro.

Pseudocódigo:

Enche a memória com registro das conversas

Estabelece heap nos registros lidos

Enquanto houver registros a serem lidos:

    Escreve topo heap em arquivo

    Lê próximo registro

    Se registro lido < topo heap

        Marca registro lido

    Insere registro no heap

    Se topo heap é marcado

        Abre novo arquivo de saída

Escreve resto do heap observando por registros marcados

Enquanto arquivos criados – arquivos lidos > 1

Abre máximo de arquivos possíveis

Lê primeiro registro guardando o arquivo de origem

Estabelece heap nos registros lidos

Abre um novo arquivo de saída

Enquanto heap não estiver vazio

Escreve topo do heap no arquivo de saída

Le próximo registro do arquivo de origem

Se não houver

Retira o topo do heap

Enquanto houver registro não lido no último arquivo criado

Lê dois registros

Enquanto os registros têm mesmo documento e palavra

Aumenta a frequência

Lê próximo um registro

Escreve os registros lidos até o diferente

Analise de Complexidade

Como em algoritmos que usam memoria secundaria o que mais leva tempo são os acessos a esta, a complexidade

temporal do algoritmo é definida pela quantidade de passadas nos arquivos.

Considerando f como o número máximo de arquivos que podem ser abertos simultaneamente, m a quantidade de registros que cabem na memória e n o número de registros a serem ordenados, o algoritmo de ordenação externa por meio de seleção por substituição realiza $P(n) = \log_f(\frac{n}{2m})$ passadas, já que em cada arquivo intermediário criado há em média 2m registros.

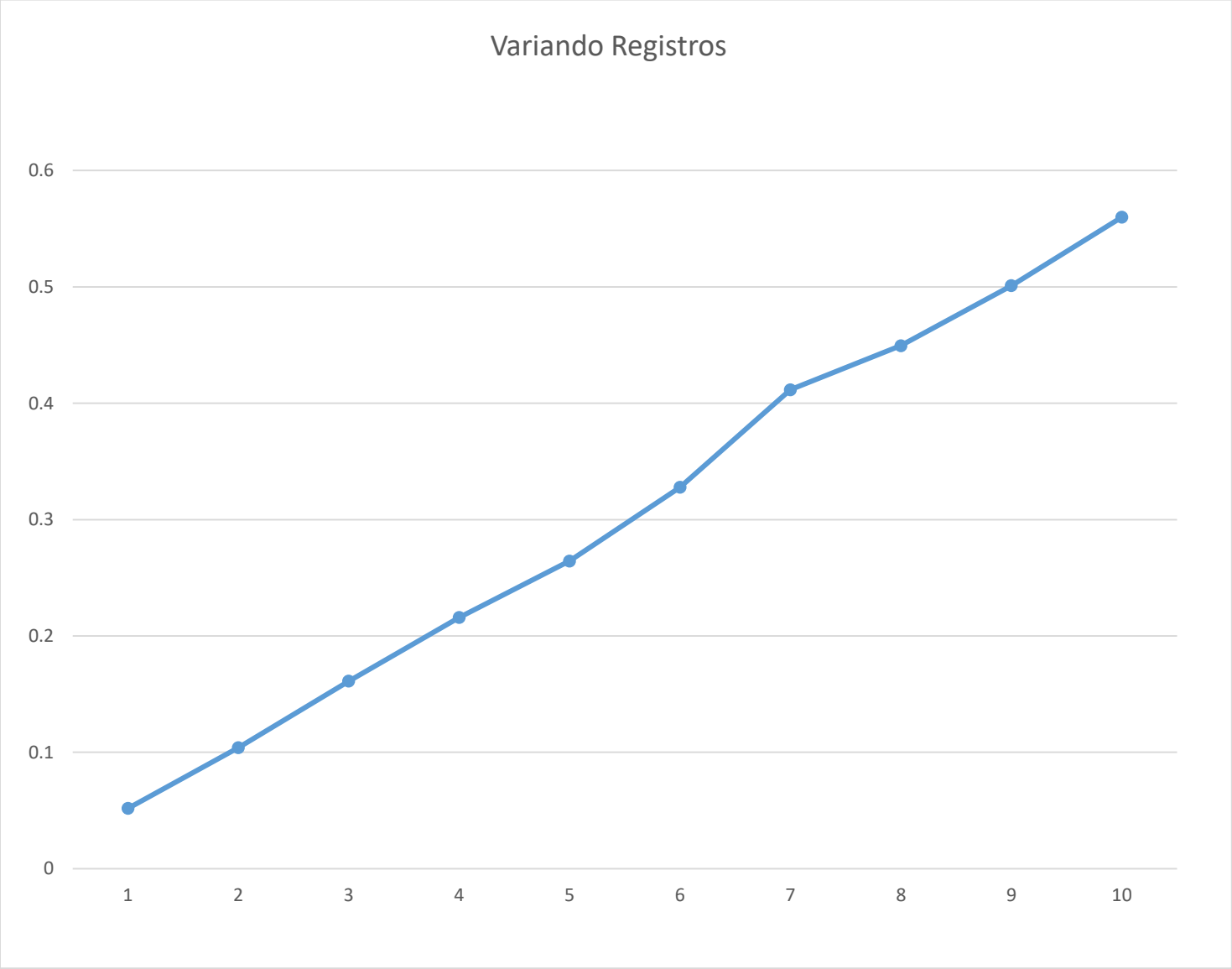
Para calcular a frequência de cada palavra realiza-se mais duas passadas no arquivo.

A estrutura que ocupa mais espaço no algoritmo é o heap que possui tamanho máximo igual a m. As outras estruturas também são limitadas por m.

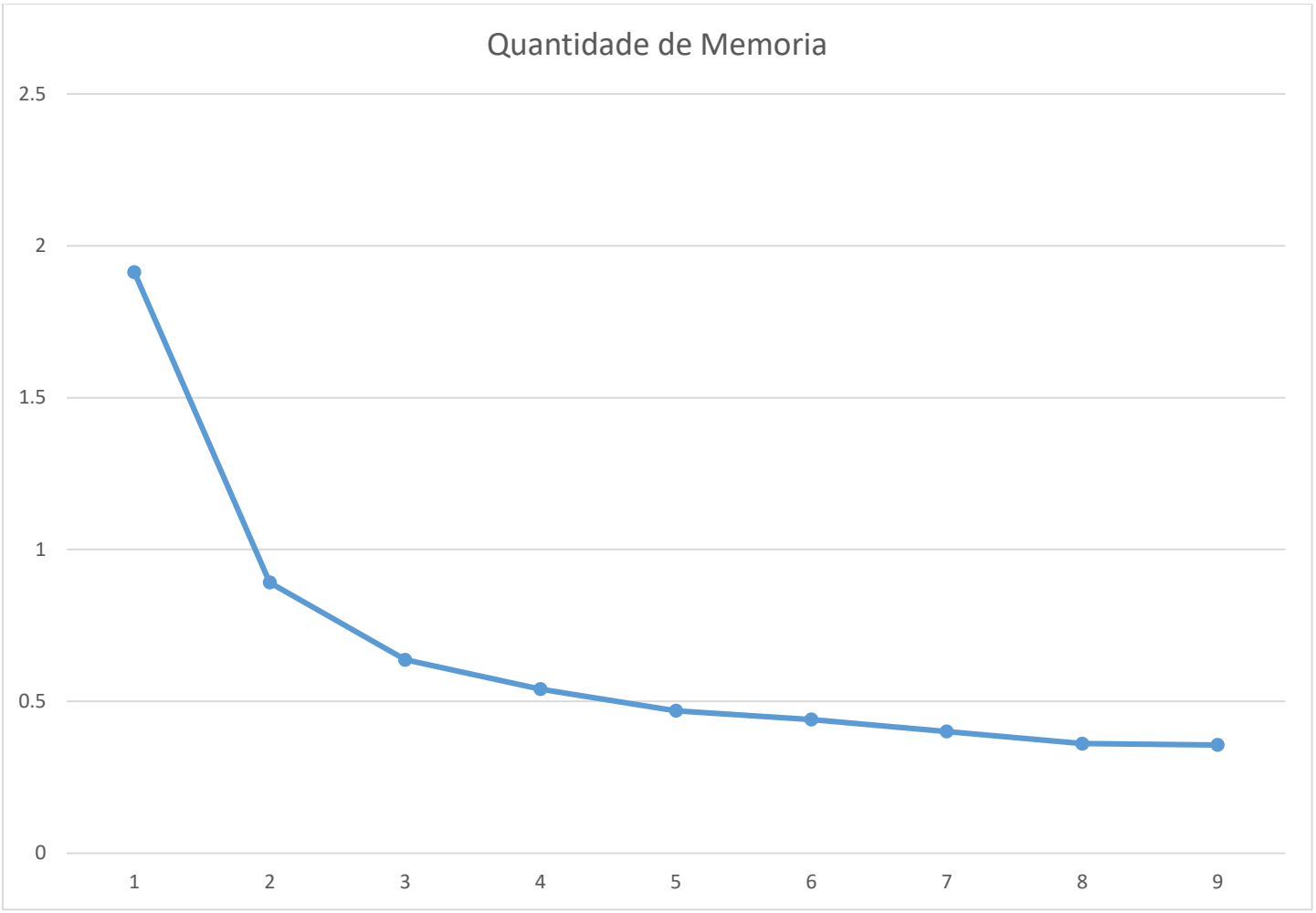
Analise Prática

Criando arquivos aleatoriamente e testando em um computador com Unbuntu e um AMD FX 6300.

Variando a quantidade de registro em 1000, começando com 1000 e mantendo a quantidade de memória constante, encontra-se o seguinte gráfico Tempo x Quantidade de registro * 1000.



Mantendo a quantidade de registros constante e fazendo a memória disponível igual a $100 \times 32 + 32 \times 10^{i}$ encontra-se o seguinte o gráfico.


Quantidade de Memoria
2.5
2
1.5
1
0.5
0
1
2
3
4
5
6
7
8
9