Trabalho Prático Aeds III

Pedro Nascimento Costa

TP1

Introdução:

A proposta apresentada pelo trabalho consiste em praticar a ideia de ordenação externa para organizar livros em estantes alfabeticamente, e, além disso, exercitar um pouco de busca binária para fazer a pesquisa por livros dentre os previamente organizados. Temos então inicialmente todos os livros que estarão presentes na biblioteca e um marcador de disponibilidade(0 indisponível, 1 disponível), além disso temos um número de estantes e a quantidade de livros que cabe em cada uma e, por fim, uma lista dos livros que se deseja pesquisar.

A ordenação externa traz o desafio de mexer com memória primária limitada, pois deseja-se ordenar algo de ordem maior do que aquilo que cabe na memória primária, esse trabalho simula tal situação. Para realizar a ordenação externa, foi utilizado um quicksort externo e para ordenar as partes carregadas na memória, a função qsort presente nas bibliotecas do C.

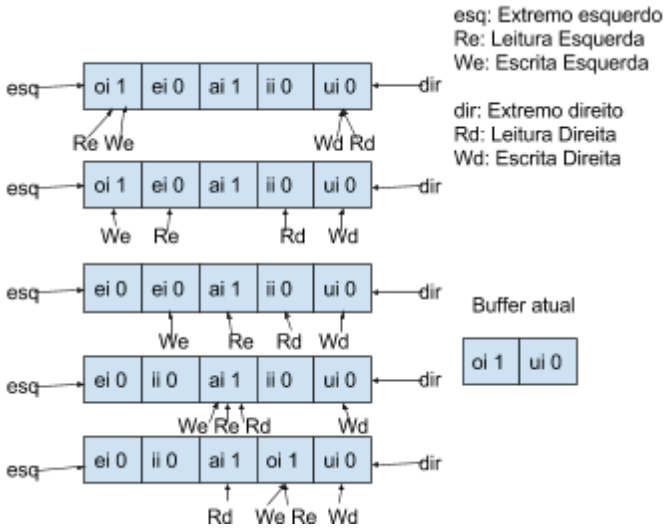


Figura I

A Figura I representa de maneira simples um exemplo de chamada do quicksort externo, no caso, o limite de memória é m = 3, então é usado m-1 espaços para o buffer e 1 para o elemento que será comparado com o buffer, além disso, o buffer representa também o pivô do algoritmo. Primeiro o buffer é montado com m-1 leituras alternadas entre esquerda e direita, segundo, usa-se ordenação interna para ordenar o buffer, terceiro, começa-se a ler de maneira alternada os demais elementos através do espaço correspondente ao elemento de comparação, então uma decisão é tomada e o elemento entra no buffer, pois está entre os extremos do mesmo, ou ele é escrito antes, ou é escrito depois do buffer. Por fim, escreve-se

o buffer ordenado e chama-se recursivamente a função para os lados a esquerda e à direita do pivô.

Desenvolvimento:

Na implementação do trabalho, foram utilizadas quatro TADs em junção com a main, são elas a TAD que administra as entradas iniciais e saídas de arquivo do programa, as TADs que realizam o quicksort externo, uma administra, a outra faz leituras e escritas e uma última TAD que trata da parte das pesquisas, fazendo também a parte da saída que não consiste em arquivos, mas sim o 'stdout'. Além disso, foi criado uma estrutura de dados para representar um livro, ela é um struct simples com um string e um char, a string contém um título e o char marca a atual disponibilidade do livro.

Para o desenvolvimento da implementação, foram separados três módulos, o primeiro realiza a administração da entrada e gera alguns arquivos iniciais e depois gera arquivos parte da saída, o segundo é a parte da ordenação externa e interna e o terceiro o das pesquisas. As TADs estão alocadas nesses módulos, mas note que elas possuem relações entre si.

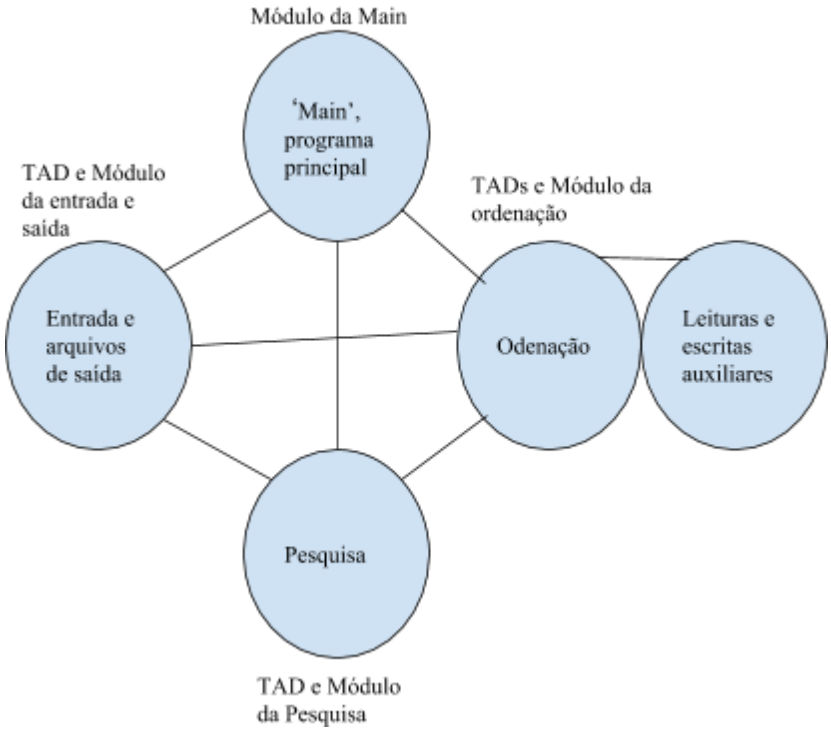


Figura II

Na Figura II temos uma representação, de maneira simples, da interação de dependência entre os componentes da implementação do programa, uma linha entre dois módulos representa que esses módulos se relacionam, por exemplo, o de pesquisa depende da criação dos arquivos com conteúdo e da ordenação dos conteúdos dos arquivos, logo a relação entre os componentes de pesquisa, entrada e arquivos de saída, e ordenação.

A entrada de dados do programa consiste primeiramente em cinco inteiros, o primeiro representa o número total de livros da biblioteca, o segundo o espaço disponível de memória

que pode ser utilizado, ou seja, o número de livros que pode estar em memória, o terceiro representa o número total de estantes, o quarto o número máximo de livros que cada estante suporta e o quinto o número de consultas/pesquisas que serão realizadas, segundamente, consiste na entrada dos livros que estão na biblioteca todos seguidos por um indicador de disponibilidade(0 indisponível, 1 disponível) e, em seguida, todos os livros a serem consultados/pesquisados. Toda a entrada é no padrão 'stdin'.

A saída consiste nos arquivos binários que representam as estantes, um arquivo de texto que representa um índice, contendo os primeiros e últimos títulos de livros de cada estante e um marcador '#' para estantes vazias, há também um arquivo de texto contendo a lista de todos os livros da biblioteca de maneira ordenada e, por fim, para cada pesquisa existe uma saída 'stdout' indicando se o livro sequer existe na biblioteca, se está disponível e caso esteja, em qual estante e posição da estante se encontra.

Os passos do programa, de maneira simples, são primeiro a criação de um arquivo binário auxiliar com todos os dados de livros da biblioteca e dos arquivos que representam as estantes, depois a ordenação do arquivo auxiliar binário, então ocorre a criação dos arquivos de texto do índice e o dos livros ordenados e o preenchimento dos arquivos binários das estantes e, por fim, são realizadas as consultas/pesquisas pelo índice e nas estantes.

Para se executar corretamente o programa, primeiro utilize o comando make, assim fazendo uso do makefile para compilar, em seguida, para executar, deve se digitar no terminal apenas: ./exec
em seguida entre com os dados de entrada, de acordo com o especificado acima.

Analisando a Complexidade:

Tempo:

As principais ações que influenciam no tempo de execução do programa são as referentes a memória secundária, como a ordenação externa, pois acessar memória secundária é significativamente mais lento.

Em relação a variação da memória disponível temos basicamente a execução do quicksort externo, que é da ordem de $O(n/b)$ no melhor caso, $O(n^2/m)$ no pior e $O((n/m)\log(n/m))$ no caso médio, sendo n o número de livros a serem ordenados, b o tamanho do arquivo sendo ordenado e m o espaço de memória que pode ser utilizado em memória primária.

Em relação ao número de estantes e suas devidas capacidades temos um 'for' que realiza no pior caso 6n acessos a arquivos, temos $O(n*a)$, sendo n o número total de livros da biblioteca e a o custo de acesso a memória secundária

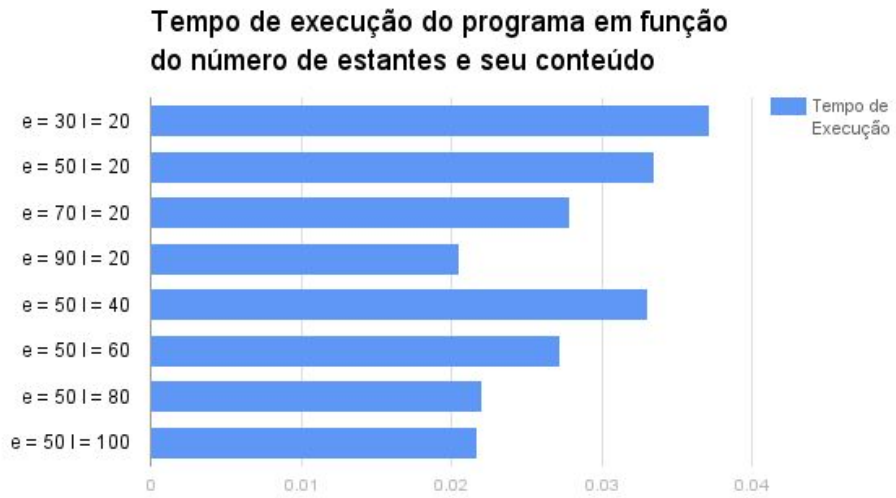
Em ralação as pesquisas temos no máximo $O(k*\log(t)*a)$, sendo t o número máximo de registros escritos no arquivo, k o número de consultas/pesquisas realizadas e a o custo de acessar um arquivo, um dado em memória secundária.

O Gráfico I abaixo representa diversos testes realizados para observar o tempo de execução da ordenação externa, nele n representa o número total de livros sendo ordenados e m o número máximo de livros que se pode carregar na memória, observa-se que quanto maior n, maior o tempo de execução e quanto menor o m, sendo o menor possível 3, maior o tempo de execução também.

Gráfico I

O Gráfico II representa o tempo de execução do programa com relação ao número de estantes e suas devidas capacidades, todo o resto é mantido constante nos testes, e representa o número de estantes e l a suas devidas capacidades. Note que mantendo o número de estantes constantes, ao aumentar suas devidas capacidades, o tempo de execução diminui e mantendo as capacidades constantes, ao aumentar o número de estantes, o tempo de execução também diminui.

Gráfico II



O Gráfico III abaixo representa o tempo de execução em função do número de pesquisas a serem realizadas, os demais elementos são mantidos constantes. k representa o número de consultas/pesquisas sendo feitas. Observe que a medida que o número de pesquisas aumenta(k aumenta), o tempo de execução também tende a aumentar.

Gráfico III

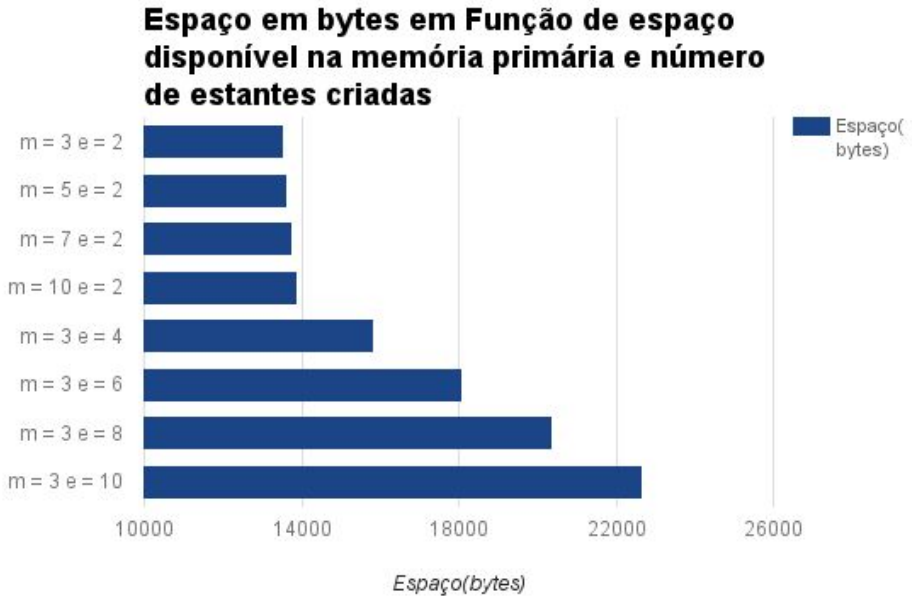
Espacial:

Em termos de espaço, o algoritmo depende basicamente do tamanho de m, o número de livros que podemos ter em memória primária e o número de estantes criadas, e. O(m+e).

Em termos de memória secundária, é difícil dizer ao certo, mas pode-se afirmar que depende dos seguintes fatores: o número de arquivos criados para as estantes e a quantidade de conteúdo de cada arquivo na saída e do arquivo auxiliar usado na implementação.

O Gráfico IV abaixo representa o uso de memória interna à medida que m aumenta, ou que e aumenta, sendo m o número de livros que pode-se ter em memória primária e e o número de estantes, os demais valores são mantidos constantes. Observe que ao aumentar m, o número de bytes aumenta de forma linear e ao aumentar e também. Além disso, percebe-se que o espaço utilizado cresce mais em função de e, do que de m.

Gráfico IV



Conclusão

A implementação teve seu maior desafio na questão da ordenação externa, nela, devido ao limite de livros que se pode ter na memória torna-se um desafio ordenar o conteúdo na memória secundária. Além disso observa-se que o custo de acesso a memória secundária torna algumas execuções mais lentas, principalmente à medida que se aumenta o número de acessos, inclusive, os principais fatores que afetam a execução, em quesitos temporais, são as operações que envolvem acessos a memória secundária.

Referências

http://stackoverflow.com/questions/5248915/execution-time-of-c-program
http://www2.dcc.ufmg.br/livros/algoritmos/cap4/slides/c/completo1/cap4.pdf
https://www.tutorialspoint.com/c_standard_library/c_function_qsort.htm
http://www.anyexample.com/programming/c/qsort__sorting_array_of_strings__integers_and_structs.xml