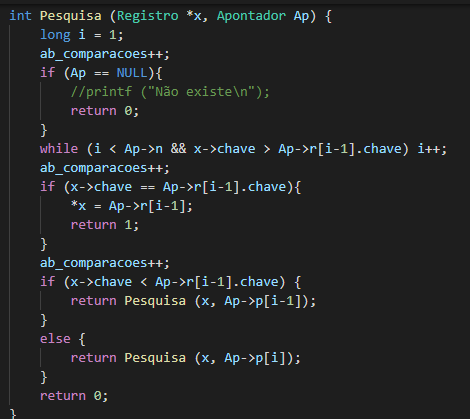
* Árvore B

1. Pesquisa em Árvore B

Para realizar a pesquisa na Árvore B, primeiro nós comparamos a chave com as outras chaves que estão na página raiz até que a chave seja encontrada ou o intervalo no qual ela se encaixa. Caso a chave não fosse localizada, o apontador iria para a subárvore do intervalo encontrado. Em seguida, o processo era repetido recursivamente até achar a chave ou atingir uma página folha (apontador nulo).

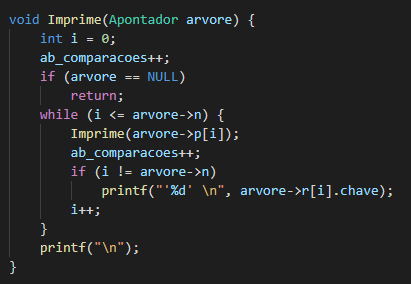
Obs: Para cada comparação, fizemos o uso de um contador.



1. Caminhamento em Árvore B

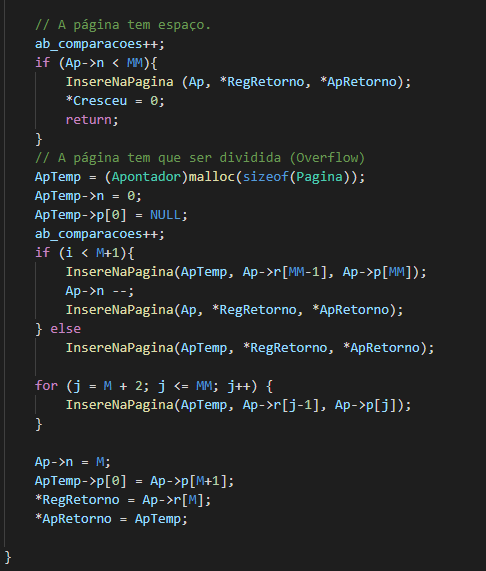
Realizamos o caminhamento de maneira recursiva, passando o apontador inicial da árvore, caso o apontador seja nulo, nada será impresso. Assim o loop será acionado, e enquanto a condição do while for aceita, cada item da primeira página vai ser selecionado, e em seguida, imprimindo seus filhos à esquerda, então todo processo é por ele (caminhamento à esquerda). Se o valor de i for diferente da quantidade de itens, então será impressa a chave daquela posição.

Obs: Para cada comparação, fizemos o uso de um contador.

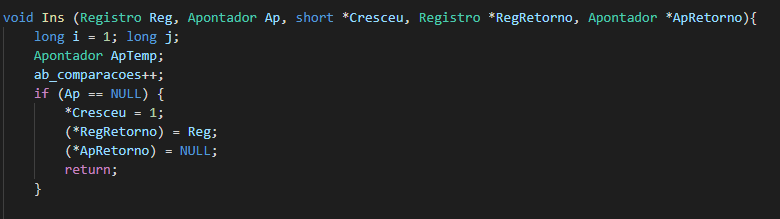


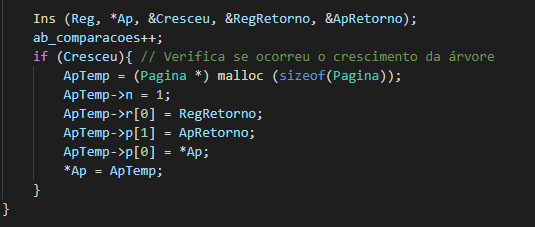
1. Inserção em Árvore B

Para o item ser inserido, primeiro nós localizamos a página onde ele deve ser inserido, caso encontre uma página válida (menos de **2m** itens), ele é inserido nela. Caso encontre uma página que esteja cheia, outra página é criada para a divisão de itens, caso a página pai também esteja cheia, a divisão se propaga.



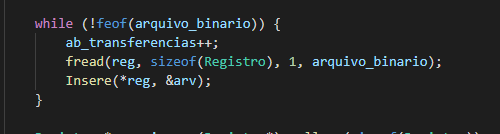
Depois, no pior caso, o processo de divisão pode propagar-se até a raiz da árvore B e, assim, sua altura aumenta. (Única forma de aumentar a altura de uma árvore B: divisão da raiz).



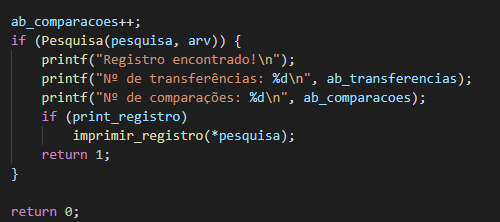


1. Execução da Árvore B

Primeiramente, fizemos um loop - que se encerra ao final do arquivo criado -, que tem a função de contar a quantidade de transferências realizadas, além de inserir os registros na árvore.



Depois, nós fazemos a pesquisa na Árvore B e criamos uma condição para quando o registro for encontrado, o usuário será avisado imprimindo uma mensagem, além da quantidade de transferências e comparações que foram realizadas durante todo o método de execução. Caso a condição não for ativada, ou seja, o registro não ser encontrado, é retornado o valor de 0.



5. Análise experimental

Para a análise experimental do método, utilizamos a função *clock* da biblioteca *time.h* para realizar o cálculo de tempo de execução dos métodos de pesquisa e variáveis inteiras para somar as comparações e leituras. Tanto as etapas de criação dos índices quanto a própria pesquisa foram contabilizadas, e as chaves pesquisadas foram determinadas por ‘*1 + rand () % (nro\_registros – 1)*’.

Os testes foram feitos alternando o registro e os arquivos (ordenado crescente, ordenado decrescente, desordenado aleatoriamente). Os outputs utilizados para a construção dos gráficos a seguir estão localizados no diretório “análise experimental” do repositório. Infelizmente, não foi possível realizar os testes de 1.000.000 de registros, pois o computador não rodou em nenhuma situação.

* Tempo de Execução

O tempo de execução aumenta consideravelmente, de forma proporcional à quantidade de registros dentro de um arquivo, entretanto, por se tratar de uma Árvore B, ela é sempre balanceada. Então, independentemente da maneira em que os registros estiverem ordenados, o tempo praticamente não se altera, considerando a mesma quantidade de registros nos 3 casos (ordenado crescente, ordenado decrescente, desordenado aleatoriamente).

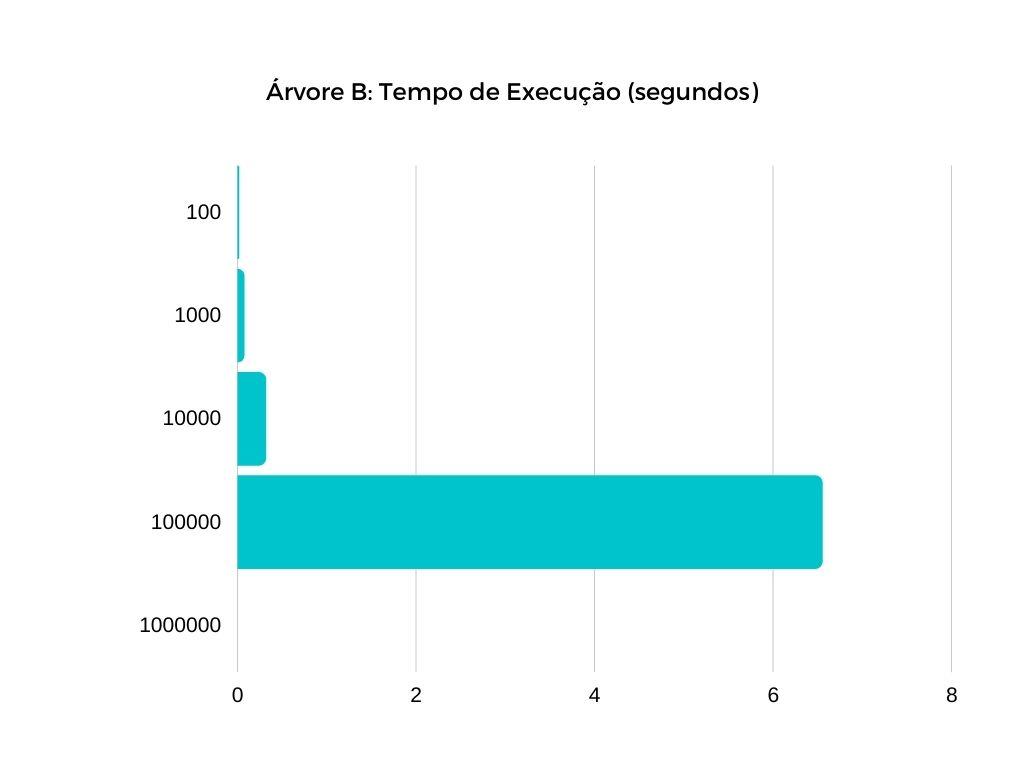


Figura 1 - Número de registros e tempo de execução (em segundos)

* Quantidade de Comparações

Assim como o tempo de execução, a quantidade de comparações aumenta consideravelmente, de forma proporcional à quantidade de registros dentro de um arquivo, entretanto, por se tratar de uma Árvore B, ela é sempre balanceada. Então, independentemente da maneira em que os registros estiverem ordenados, o número de comparações praticamente não se altera, considerando a mesma quantidade de registros nos 3 casos (ordenado crescente, ordenado decrescente, desordenado aleatoriamente).

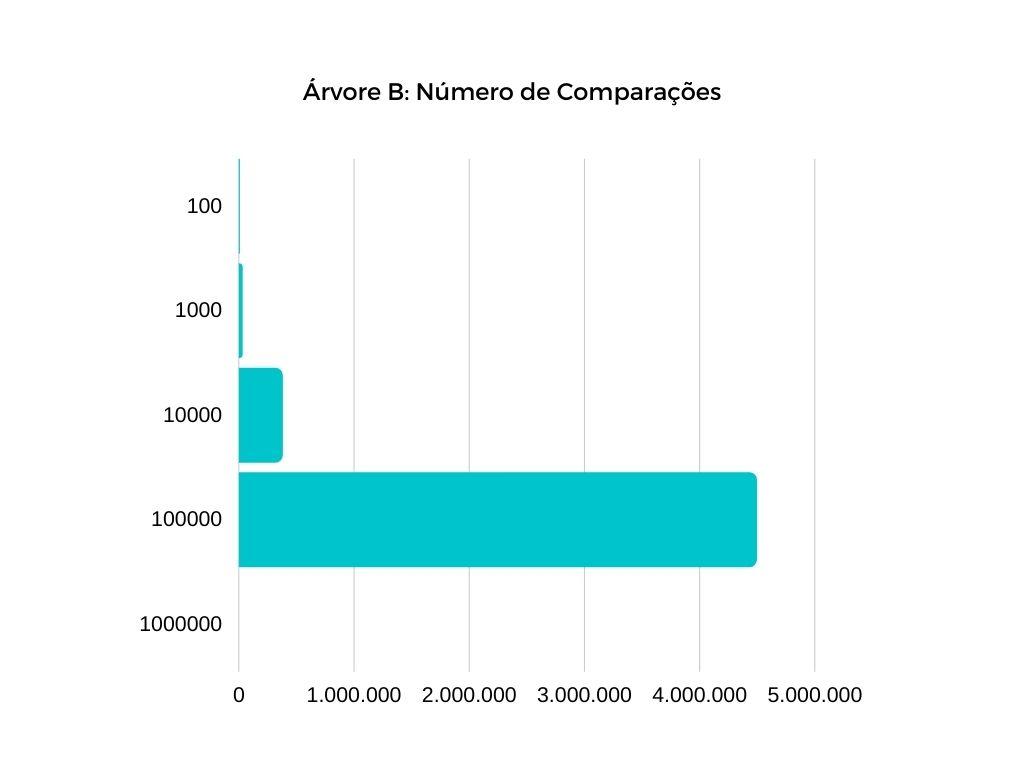


Figura 2 - Número de registros e número de comparações realizadas

* Quantidade de Transferências
* O número de transferências é sempre constante, independentemente da maneira do teste, ela sempre será igual a quantidade de registros na Árvore B.

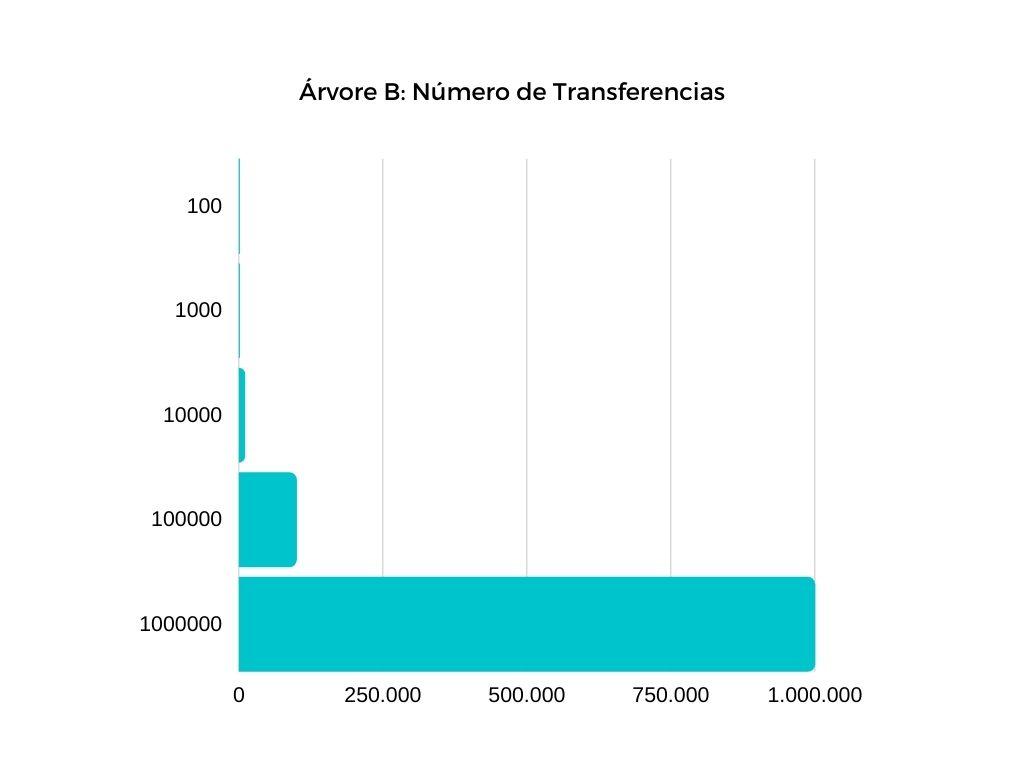


Figura 3 - Número de registros e número de transferências realizadas