4. Conclusões *(as linhas no template representam a extensão máxima de texto manuscrito)*

4.1 Tarefa 1

A árvore binária Tem a maior complexidade para as operações de pesquisa e inserção. No pior caso, inserir requer pesquisar todos os elementos. Sendo assim, inserir N chaves tem uma complexidade de $O(n \cdot (n+1)/2)$ (Soma de Gauss), ou aproximadamente $O(n^2)$. A regressão feita no gráfico correspondente suporta esta conclusão.

Por outro lado, a natureza implícita desta estrutura e a falta de ordenação permite remover elementos em $O(1)$ e N elementos em $O(N)$.

4.2 Tarefa 2

A árvore AVL Tem o melhor balanceamento por natureza, e logo deveria realizar ~~rotações~~ um maior número de rotações que os restantes. Para conjuntos pequenos de keys esta overhead pode não valer a pena, mas para conjuntos maiores o balanceamento mais estrito resulta em operações de pesquisa mais eficientes.

A AVL mantém uma complexidade de tempo de $O(\log_2 N)$ consistente para as suas pesquisas.

4.3 Tarefa 3

A árvore Red-Black é menos estritamente balanceada que a AVL, o que é benéfico para conjuntos com tamanhos mais pequenos e tamanhos médios. O menor número de rotações significa que é mais rápido inserir em média, mas à medida que o balanceamento vai diminuindo, esta vantagem pode desaparecer, o que aparenta ser o caso nas ~~outras~~ medidas tomadas.

A operação de pesquisa ~~tipicamente~~ pode perder eficiência quando o balanço diminuir.