Bruno Baffini Santana - 11834050 e Ilan Matheus Silva de Sousa - 11832920

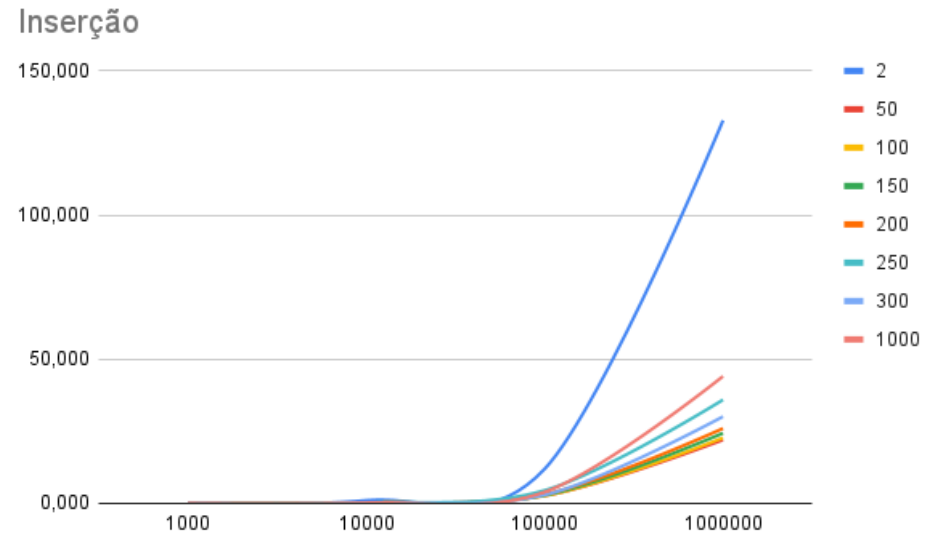
Todo o código foi montado a partir do modelo enviado pelo professor, as funções de inserção e busca foram implementadas a partir do pseudo-código disponibilizado nos slides do professor. A remoção foi implementada com os casos também passados nos slides do professor, porém nós desenvolvemos toda a lógica e funções implementadas.

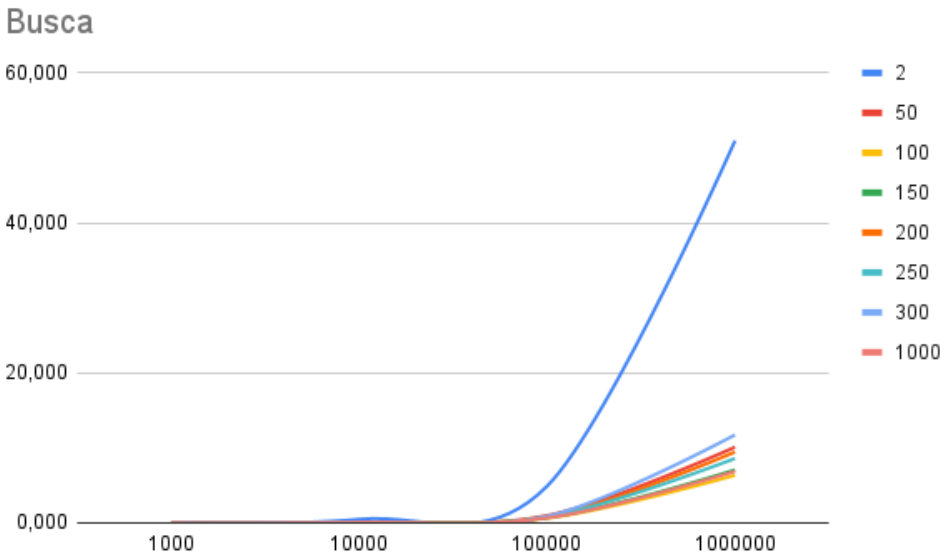
Macbook Pro (13-inch, Mid 2012)
Processador 2,5 Ghz Intel Core i5
Memória 8Gb 1600Mhz DDR3
Memória secundária SSD SATA 240Gb

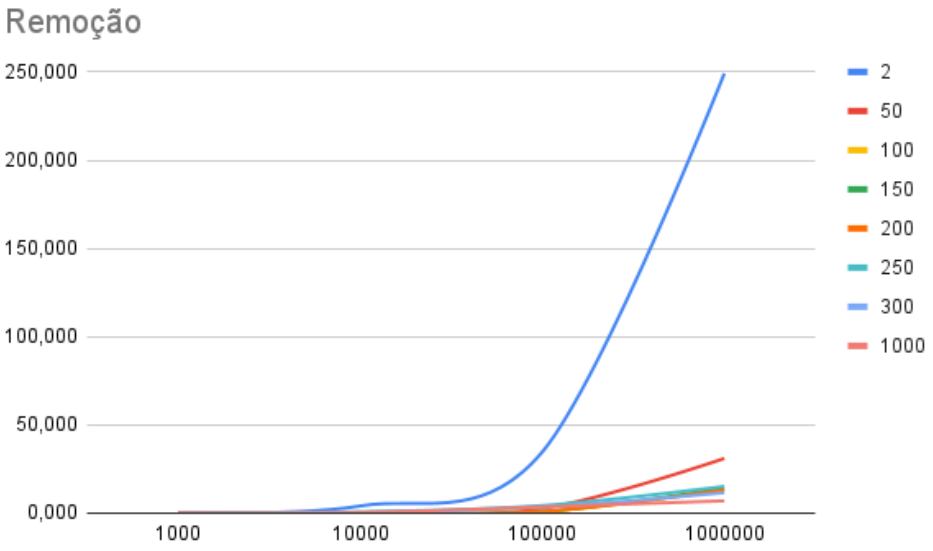
Inserção								
Quantidade\t	2	50	100	150	200	250	300	1000
100	0,138	0,018	0,009	0,028	0,015	0,010	0,010	0,037
1000	1,033	0,223	0,237	0,257	0,217	0,213	0,190	0,208
10000	11,941	2,666	2,633	2,806	3,144	4,507	2,913	3,936
100000	132,852	22,040	22,730	24,357	25,996	35,931	30,111	44,086
1000000	1207,299	244,602	252,240	268,492	270,814	396,772	289,950	494,353

Busca								
Quantidade\t	2	50	100	150	200	250	300	1000
100	0,045	0,009	0,006	0,007	0,007	0,007	0,011	0,007
1000	0,481	0,100	0,059	0,069	0,080	0,077	0,096	0,064
10000	4,898	0,971	0,623	0,688	0,787	0,785	0,877	0,658
100000	50,970	10,102	6,356	7,065	9,461	8,596	11,741	6,866
1000000	522,491	108,622	63,525	69,393	85,802	80,384	97,638	68,114

Remoção								
Quantidade\t	2	50	100	150	200	250	300	1000
100	0,452	0,051	0,038	0,103	0,030	0,093	0,045	0,050
1000	4,061	0,149	0,277	0,320	0,323	0,753	0,373	0,328
10000	34,628	2,619	0,780	0,841	0,847	4,283	3,441	3,565
100000	249,098	30,978	13,104	14,210	13,265	15,156	11,581	6,949
1000000	1311,196	213,691	483,913	164,788	189,840	174,221	187,210	68,376


Inserção
150,000
100,000
50,000
0,000
1000
10000
100000
1000000
2
50
100
150
200
250
300
1000


Busca
60,000
40,000
20,000
0,000
1000
10000
100000
1000000
2
50
100
150
200
250
300
1000


Remoção
250,000
200,000
150,000
100,000
50,000
0,000
1000
10000
100000
1000000
2
50
100
150
200
250
300
1000

Após analisar os casos de teste, foi possível perceber que para casos com poucas inserções, buscas e remoções a variação do valor *t* das árvores não foi significativa o bastante para causar uma diferença nos tempos de execução. A partir de 10.000 execuções foi possível notar uma discrepância no tempo de execução para a árvore com t = 2, essa discrepância no tempo de execução aumentou significativamente para o caso com 1.000.000 de testes, tornando inviáveis testes com um número maior de execuções. Foi possível perceber também que a árvore com t = 1000 não era tão eficiente para os valores testados por possuir um valor máximo de chaves grande o bastante para fazer com que a estrutura das árvores B não fizesse mais sentido (para parte dos casos de teste trabalhar com a árvore B de t = 1000 era quase equivalente a trabalhar com arranjos). Outra observação é que as operações de inserção, busca e remoção possuem gráficos muito semelhantes, apesar da operação de busca ser a menos custosa, seguida pela inserção e com a remoção possuindo um maior custo de tempo, as relações entre as médias para cada *t* se mantém as mesmas.

Concluindo, é necessário um conhecimento prévio do número de elementos em uma árvore B para definir o valor ideal de *t*, este sendo o mais eficiente para todos os tipos de operações na árvore B.