In a couple of paragraphs, explain your strategy for comparing the timing of the two approaches. How many elements did you insert? How many times did you perform the simulation?

Answer: In the bstMain.c, I use srand(2) to generate the totally same sequence of value for two algorithm. then from the number of 10000 values to 160000 values to add into the tree. Every time the number of elements for insert double until it reaches 160000. use a two-dimensional array to store the values in order to do the remove. using rand%n to generate the random values. Then perform the remove function.

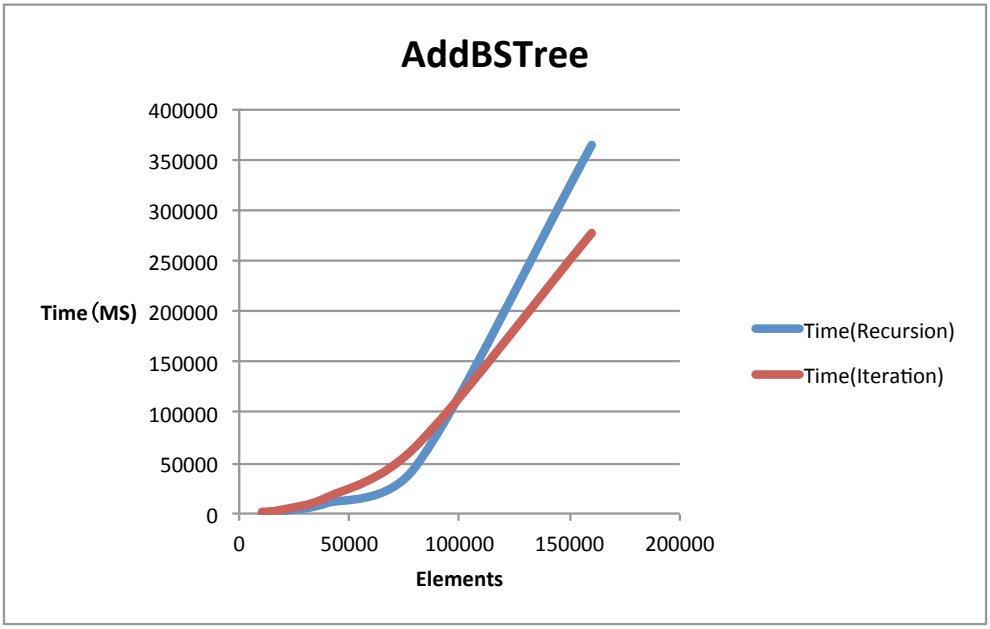
Which BST is faster and why do you think this is the case?

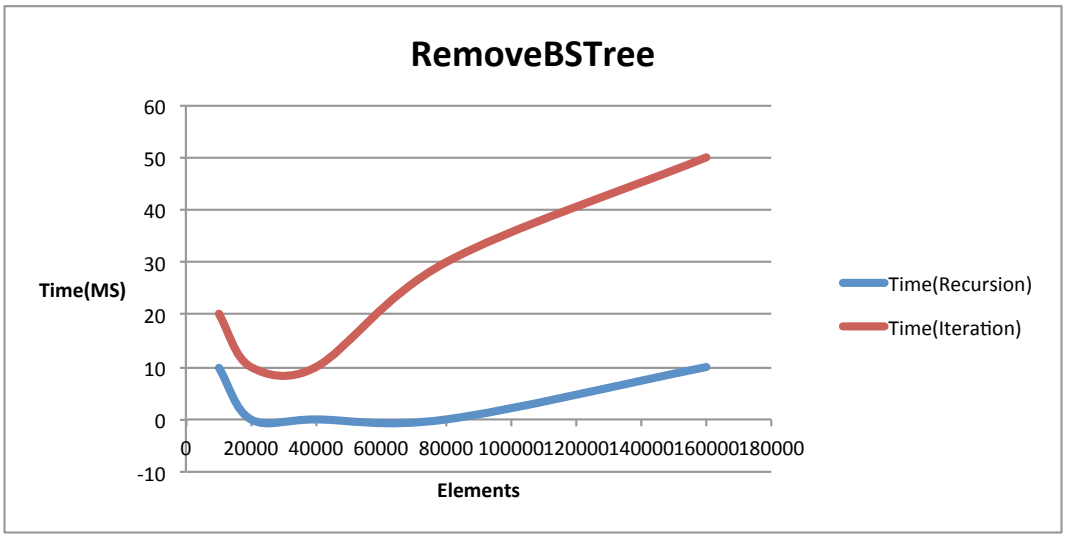
Answer: For this case, the two algorithms have nearly same speed at beginning. when the number of elements increase more, the Iterative BTS will faster then Recursive BST.

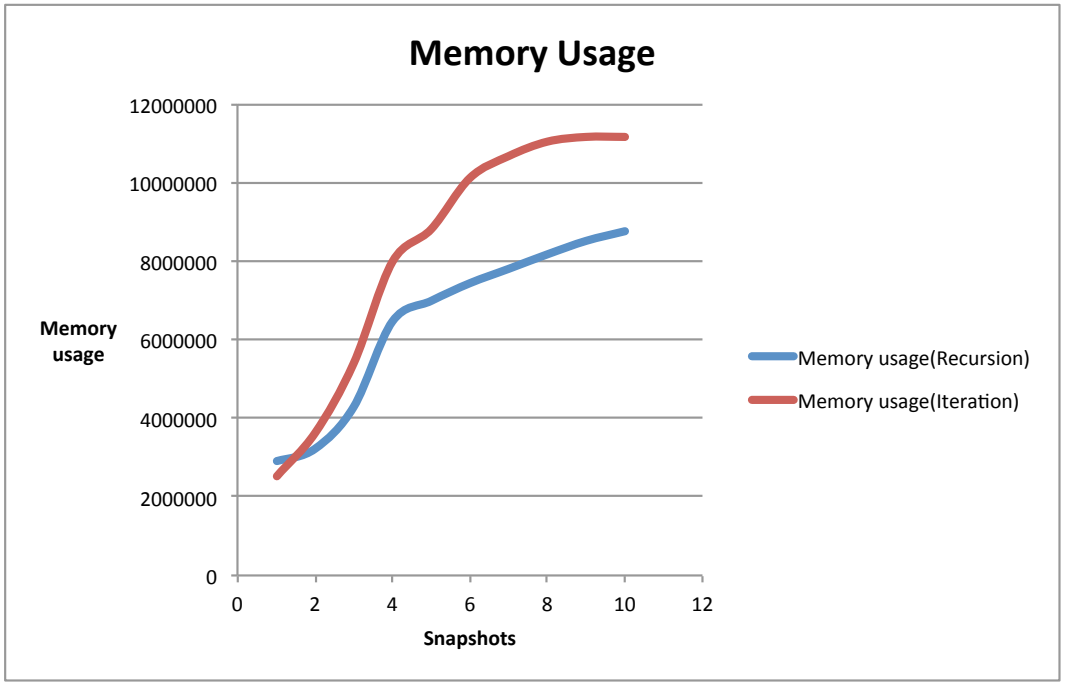
Which BST takes up less memory? Explain why.

Answer: For this case, the result shows Recursive BST cost less memory. But in general, the Recursive BST should cost more memory because we need to perform many functions in Recursive BST, So there are a lot of extra memory cost in the stack.

Create and turn in two charts graphing the comparison of the two BST implementations in terms of time, and memory usage (See Assignment 3 comparisons).




RemoveBSTree
60
50
40
30
20
10
0
-10
Time(MS)
0
20000
40000
60000
80000
100000
120000
140000
160000
180000
Elements
Time(Recursion)
Time(Iteration)


Memory Usage
12000000
10000000
8000000
6000000
4000000
2000000
0
Memory usage
0
2
4
6
8
10
12
Snapshots
Memory usage(Recursion)
Memory usage(Iteration)