Task 4: Benchmark search() on your different variations of splay trees on the four different access patterns provided.

Time is written in milliseconds

Splay Tree Variants	Access Pattern				
	Sequential	Uniform	Skewed	Working Set	
Bottom-up	99999	1794831	986319	1809486	Total rotation count
	4.41426	21.1883	19.2143	21.3063	Average search depth
	864.688	807.8021	926.3291	988.85	Total time
Top-Down	98,000	586446	322929	97987	Total rotation count
	4.31964	21.3205	19.333	4.59037	Average search depth
	809.3626	847.5102	830.792	868.3314	Total time
Semi-splay (option 1)	688088	585930	322513	590380	Total rotation count
	22.0983	21.3251	19.3345	21.4419	Average search depth
	1,021.49	928.21	1,091.62	1,022.10	Total time
Weighted splay (option 1)	688088	585930	322513	590380	Total rotation count
	22.2297	21.249	19.2475	21.5304	Average search depth
	1,069.20	993.77	1,072.81	1,227.27	Total time

Task 5: Answer these questions briefly and clearly

- Is there an asymptotic bound difference in the complexity of the top-down and bottom-up implementations? What are the practical differences?

>> There is no asymptotic bound difference in the complexity of the top down and bottom up implementations because for both the top down and bottom up approach the amortized time complexity is O(log n) and the worst case time complexity is O(n). So, the only difference between top down and bottom up is the way we implement them. During bottom up the accessed nodes moves up using the parent pointers and does the rotations to them when going towards the root. But, Top down splaying restructure the entire tree when it is going down during the searching of the node itself. So, Top down implementation can be more efficient as it requires less pointer adjustments.

- Why is a splay tree not thread-safe?

>> Every operation performed in the splay tree modifies the tree due to various rotations performed. So, if two or more threads attempt to access or modify the splay tree at the same time it can cause the splay tree to corrupt or crash. So, splay tree is not thread safe.

- What is the advantage of semi-splay and weighted-splay respectively.

>> The advantage of semi-splay is that it reduced the number of rotations performed during each access. Instead of completely splaying a node to the tree, semi splays limits the complete splaying by performing only a fixed number of rotations. This can help to improve the performance of the program by reducing the cost per operation.

>> The advantage of weighted splay is that it makes frequently accessed nodes remain near the top of the tree and makes it faster to access them. In weighted splay each node has a weight that represents how many time it has been accessed. So, this allows the tree to have frequently accessed nodes remain near the top of the tree.

- Explain your observations from Task 4 in a detailed write-up.

When observing the above table, we can see that Top down splay tree has the best performance as compared to other trees as the time taken for it is comparatively less as compared to other trees. The Bottom up splay tree has had more rotations as compared to Top down as seen on the table. Also the average search depth for bottom up and top down is nearly same for most of the access patterns. Whereas, Semi splay and weighted splay both had higher average search depth as compared to Bottom up and Top down splay trees. The total rotation count as well as total time taken to compile is also nearly similar for both the Semi splay and Weighted splay. We can see Top down splay tree performs better as compared to Bottom up splay because it restructures the tree during the initial search itself. We can also observe both the Semi splay and Weighted splay perform worse than expected due to their higher search depth and execution time.