Run 1: cap =8

```
clang++-7 -pthread -std=c++17 -o main BinarySearchTree.cpp NonSortedArray.cpp SortedArray.cp
./main
The content of the test case vector:
5 29 9 25 1 21 13 17
Adding 5
Sorted Array        0 comparisons       0 shifts
Adding 29
Sorted Array        0 comparisons       0 shifts
Adding 9
Sorted Array        1 comparisons       1 shifts
Adding 25
Sorted Array        1 comparisons       1 shifts
Adding 1
Sorted Array        4 comparisons       4 shifts
Adding 21
Sorted Array        2 comparisons       2 shifts
Adding 13
Sorted Array        3 comparisons       3 shifts
Adding 17
Sorted Array        3 comparisons       3 shifts


1
5
9
13
17
21
25
29
Values of the sorted array: 1 5 9 13 17 21 25 29
Values of the non sorted array: 5 29 9 25 1 21 13 17

Searching for 11
Sorted Array        4 comparisons   not found
Non Sorted Array    8 comparisons   not found
Searching for 4
Sorted Array        3 comparisons   not found
Non Sorted Array    8 comparisons   not found
Searching for 7
Sorted Array        3 comparisons   not found
Non Sorted Array    8 comparisons   not found
Searching for 5
Sorted Array        2 comparisons   found
Non Sorted Array    1 comparisons   found
```

```
Adding 35
Adding 18
Adding 14
Before operator =, bstCopy:
14
18
35
In operator=, first release the nodes in the invoking object:
Delete: 14
Delete: 18
Delete: 35

Now copy the nodes from the parameter object:
Created node:   5
Created node:   1
Created node:   29
Created node:   9
Created node:   25
Created node:   21
Created node:   13
Created node:   17

After =, adding 50 to the copy, Adding 85 to the original, After updating,
1
5
9
13
17
21
25
29
85
the copy:
1
5
9
13
17
21
25
29
50
```

```
End of the program is reached. The BSTs are to be destructed.
Releasing nodes in the order of:
Delete: 1
Delete: 17
Delete: 13
Delete: 21
Delete: 25
Delete: 9
Delete: 50
Delete: 29
Delete: 5

Releasing nodes in the order of:
Delete: 1
Delete: 17
Delete: 13
Delete: 21
Delete: 25
Delete: 9
Delete: 85
Delete: 29
Delete: 5
```

Run 2 cap = 9

```
The content of the test case vector:
25 1 17 9 5 21 29 13 33
Adding 25
Sorted Array        0 comparisons       0 shifts
Adding 1
Sorted Array        1 comparisons       1 shifts
Adding 17
Sorted Array        1 comparisons       1 shifts
Adding 9
Sorted Array        2 comparisons       2 shifts
Adding 5
Sorted Array        3 comparisons       3 shifts
Adding 21
Sorted Array        1 comparisons       1 shifts
Adding 29
Sorted Array        0 comparisons       0 shifts
Adding 13
Sorted Array        4 comparisons       4 shifts
Adding 33
Sorted Array        0 comparisons       0 shifts


1
5
9
13
17
21
25
29
33
Values of the sorted array: 1 5 9 13 17 21 25 29 33
Values of the non sorted array: 25 1 17 9 5 21 29 13 33

Searching for 8
Sorted Array        3 comparisons   not found
Non Sorted Array    9 comparisons   not found
Searching for 12
Sorted Array        4 comparisons   not found
Non Sorted Array    9 comparisons   not found
```

```
Adding 10
Adding 16
Adding 12
Before operator =, bstCopy:
10
12
16
In operator=, first release the nodes in the invoking object:
Delete: 12
Delete: 16
Delete: 10

Now copy the nodes from the parameter object:
Created node:   25
Created node:   1
Created node:   17
Created node:   9
Created node:   5
Created node:   13
Created node:   21
Created node:   29
Created node:   33

After =, adding 50 to the copy, Adding 85 to the original, After updating, the original:
1
5
9
13
17
21
25
29
33
85
the copy:
1
5
9
13
17
21
25
29
33
50
```

```
End of the program is reached. The BSTs are to be destructed.
Releasing nodes in the order of:
Delete: 5
Delete: 13
Delete: 9
Delete: 21
Delete: 17
Delete: 1
Delete: 50
Delete: 33
Delete: 29
Delete: 25

Releasing nodes in the order of:
Delete: 5
Delete: 13
Delete: 9
Delete: 21
Delete: 17
Delete: 1
Delete: 85
Delete: 33
Delete: 29
Delete: 25

❯
```

Run 3: cap =7

```
13 7 11 1 3 9 5
Adding 13
Sorted Array        0 comparisons       0 shifts
Adding 7
Sorted Array        1 comparisons       1 shifts
Adding 11
Sorted Array        1 comparisons       1 shifts
Adding 1
Sorted Array        3 comparisons       3 shifts
Adding 3
Sorted Array        3 comparisons       3 shifts
Adding 9
Sorted Array        2 comparisons       2 shifts
Adding 5
Sorted Array        4 comparisons       4 shifts


1
3
5
7
9
11
13
Values of the sorted array: 1 3 5 7 9 11 13
Values of the non sorted array: 13 7 11 1 3 9 5

Searching for 3
Sorted Array        2 comparisons   found
Non Sorted Array    5 comparisons   found
Searching for 12
Sorted Array        3 comparisons   not found
Non Sorted Array    7 comparisons   not found
Searching for 1
Sorted Array        3 comparisons   found
Non Sorted Array    4 comparisons   found
Searching for 1
Sorted Array        3 comparisons   found
Non Sorted Array    4 comparisons   found
Searching for 11
Sorted Array        2 comparisons   found
Non Sorted Array    3 comparisons   found
Searching for 11
Sorted Array        2 comparisons   found
Non Sorted Array    3 comparisons   found
```

```
Adding 18
Adding 12
Adding 15
Before operator =, bstCopy:
12
15
18
In operator=, first release the nodes in the invoking object:
Delete: 15
Delete: 12
Delete: 18

Now copy the nodes from the parameter object:
Created node:   13
Created node:   7
Created node:   1
Created node:   3
Created node:   5
Created node:   11
Created node:   9

After =, adding 50 to the copy, Adding 85 to the original, After updating, the original:
1
3
5
7
9
11
13
85
the copy:
1
3
5
7
9
11
13
50
```

```
End of the program is reached. The BSTs are to be destructed.
Releasing nodes in the order of:
Delete: 5
Delete: 3
Delete: 1
Delete: 9
Delete: 11
Delete: 7
Delete: 50
Delete: 13

Releasing nodes in the order of:
Delete: 5
Delete: 3
Delete: 1
Delete: 9
Delete: 11
Delete: 7
Delete: 85
Delete: 13
```

In your own words, describe the pros and cons of using binary search trees to organize data.

The pros are the time complexity of insert, delete, and search is logarithmic. That is very inportant when there are a lot of nodes.

Traversal is always the straightforward and predictable because of the ordering of nodes.

We can look at the BST as an array, so finding the smallest and largest element is easy.

The disadvantage is that the tree can be linear and it can be hard to implement a balanced tree.

*Something I noticed during the cap 31 runs is that the comparisons for unsorted is so high.