

Yes from calculating the number of operatations it takes to find the target value K, the two algortims could be compared. Multiple differnet sets of data where run through the program to test this.

By looking at this graph when the input of the function is small, the BFS\_subset\_Sum algorithm should technically run faster, but for large majority of the graph the HS\_Subset\_Sum would run faster. This can be proven by comparing the amount time taken to the num of operations its takes to compute the final solutions.

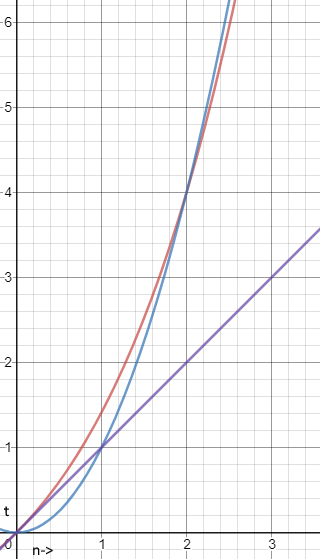
The data gathered from the test values of S, can be seen in the table below.

Figure 1 The Blue line is O(2^n) The Red line is O(n\*2^(n/2)) and the purple line is O(n)

# Number of operations to find K

|  |  |  |
| --- | --- | --- |
| BFI\_SUBSET #Of Operations | HS\_SUBSET #Of Operations | Set S (searching for k= 10) |
| 4 | 3 | 13 19 7 13 16 20 2 10 6 11 6 12 |
| 28 | 23 | 5 3 3 18 13 5 10 10 12 19 |
| 15 | 15 | 8 1 13 12 19 2 4 9 12 2 1 19 16 9 |
| 0 | 0 | 11 15 2 10 3 12 19 12 19 16 13 6 7 6 1 |
| 16 | 7 | 18 16 17 6 20 6 8 15 9 17 2 8 13 1 |
| 3 | 6.7 | 15 1 3 9 |
| 10 | 23 | 8 19 1 3 6 18 |

When the number of elements in the array S is big then HS\_SUBSET runs the fastest but if the number of elements is a small value then BFI\_SUBSET runs faster. By comparing this theory with the above graph, it can be shown that the prediction is correct.