Implement Search On BST:

You are given a Binary Search Tree (BST) where each node is represented as a dictionary with keys "value", "left", and "right". The "value" key holds the node's value, while "left" and "right" keys hold the left and right subtrees, respectively, which are also represented as dictionaries or None if there are no children.

Your task is to search for a target value in the BST. If the target value exists in the BST, return the subtree rooted at the node with that value. If the target value does not exist, return None.

Input Format:

- A dictionary representing the root of the BST.
- An integer target representing the value to be searched in the BST.

Output Format:

- A dictionary representing the subtree rooted at the node with the target value if it exists, or None if the target value is not found. The output should be prefixed with "Result: ".

Example:

Input:

```
bst_input = {
    "value": 4,
    "left": {
        "value": 2,
        "left": {"value": 1, "left": None, "right": None},
        "right": {"value": 3, "left": None, "right": None}
    },
    "right": {
        "value": 7,
        "left": None,
        "right": None
    }
}
```

2

Output:

{'value': 2, 'left': {'value': 1, 'left': None, 'right': None}, 'right': {'value': 3, 'left': None, 'right': None}}

Explanation:

The BST is represented as a dictionary. The function searches for the target value 2 in the BST. It finds the node with value 2 and returns the subtree rooted at that node.