Joseph Edradan

4/18/19

CIS 279 & CIS 256

**Assignment #3-A: Binary Trees and AVL Trees**

Exercise 1: Determine the Order



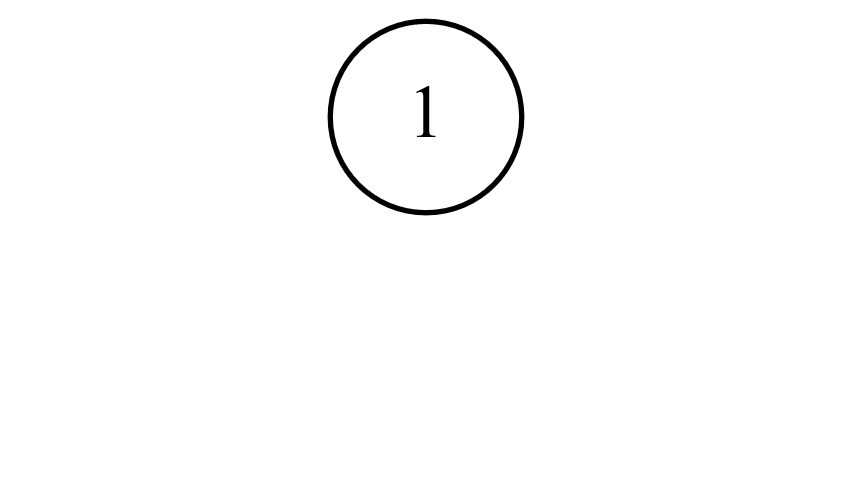
*Pre-Order* – A, H, G, I, F, E, B, C, D

*In-Order* – G, H, F, I, E, A, B, D, C

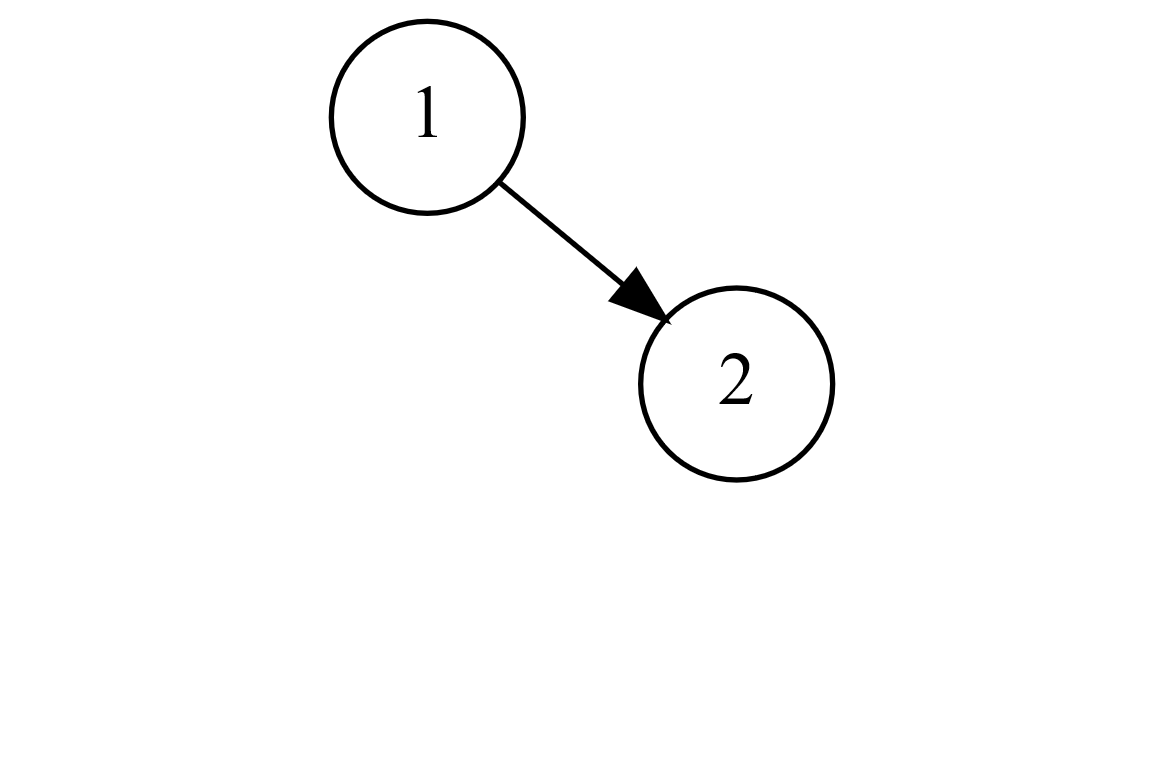
*Post-Order* – G, F, E, I, H, D, C, B, A

Exercise 2: Insert and Rotate

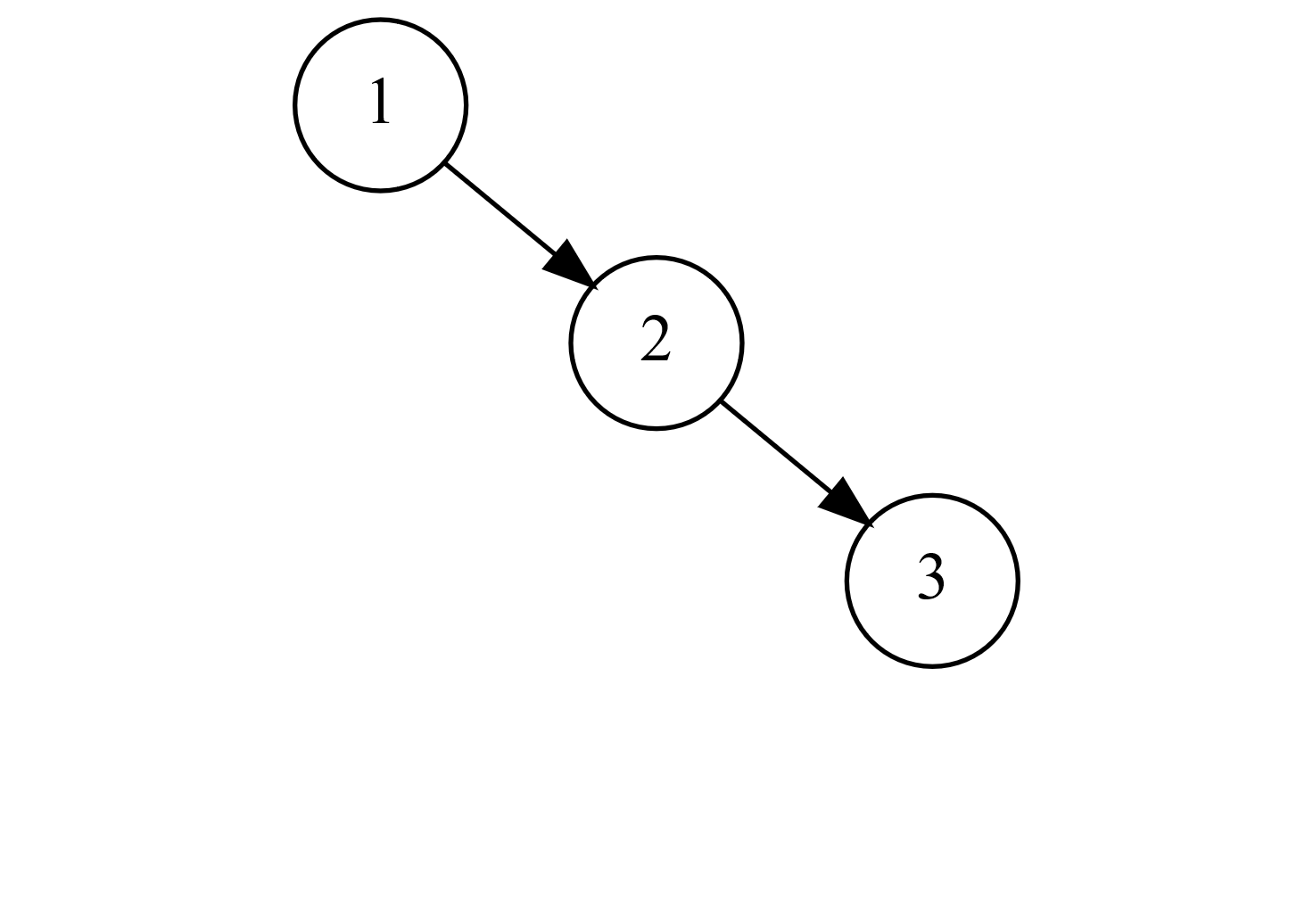
Insert 1



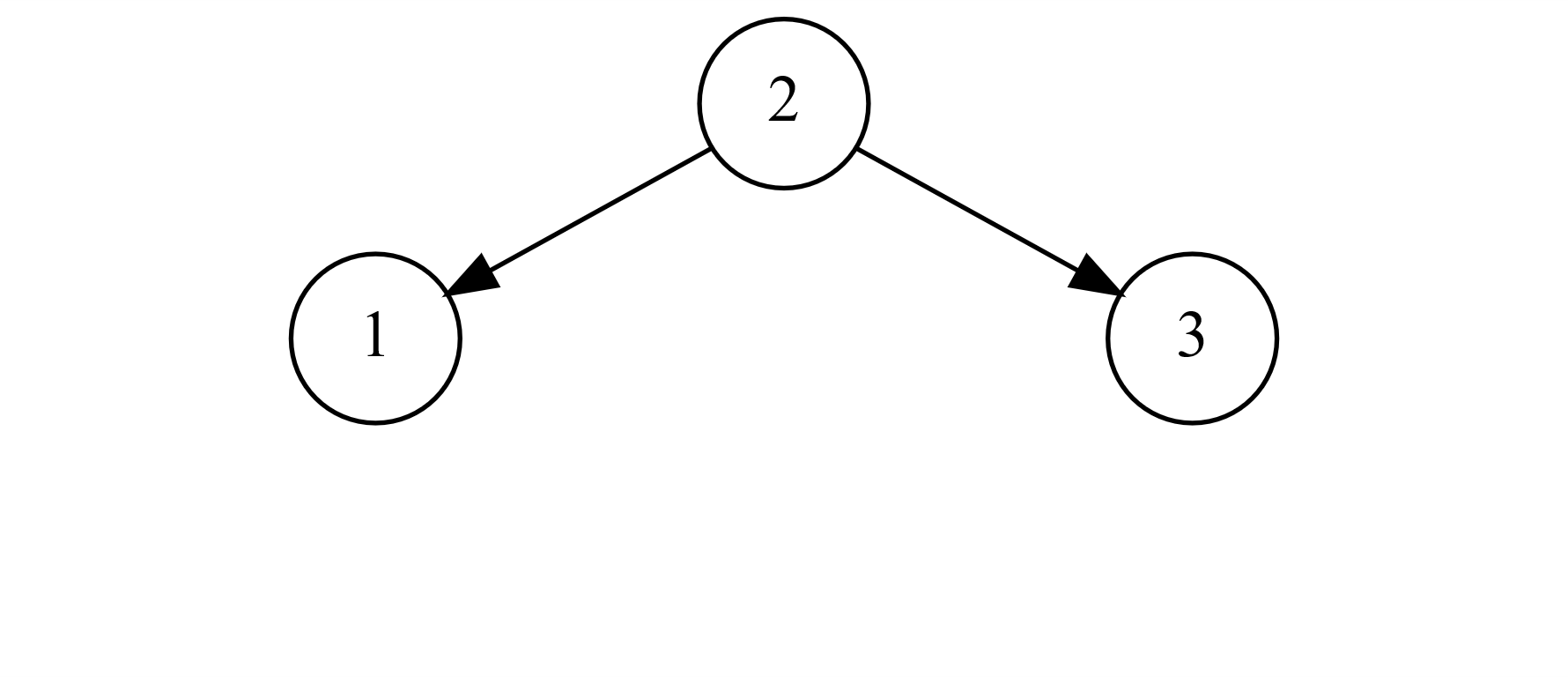
Insert 2



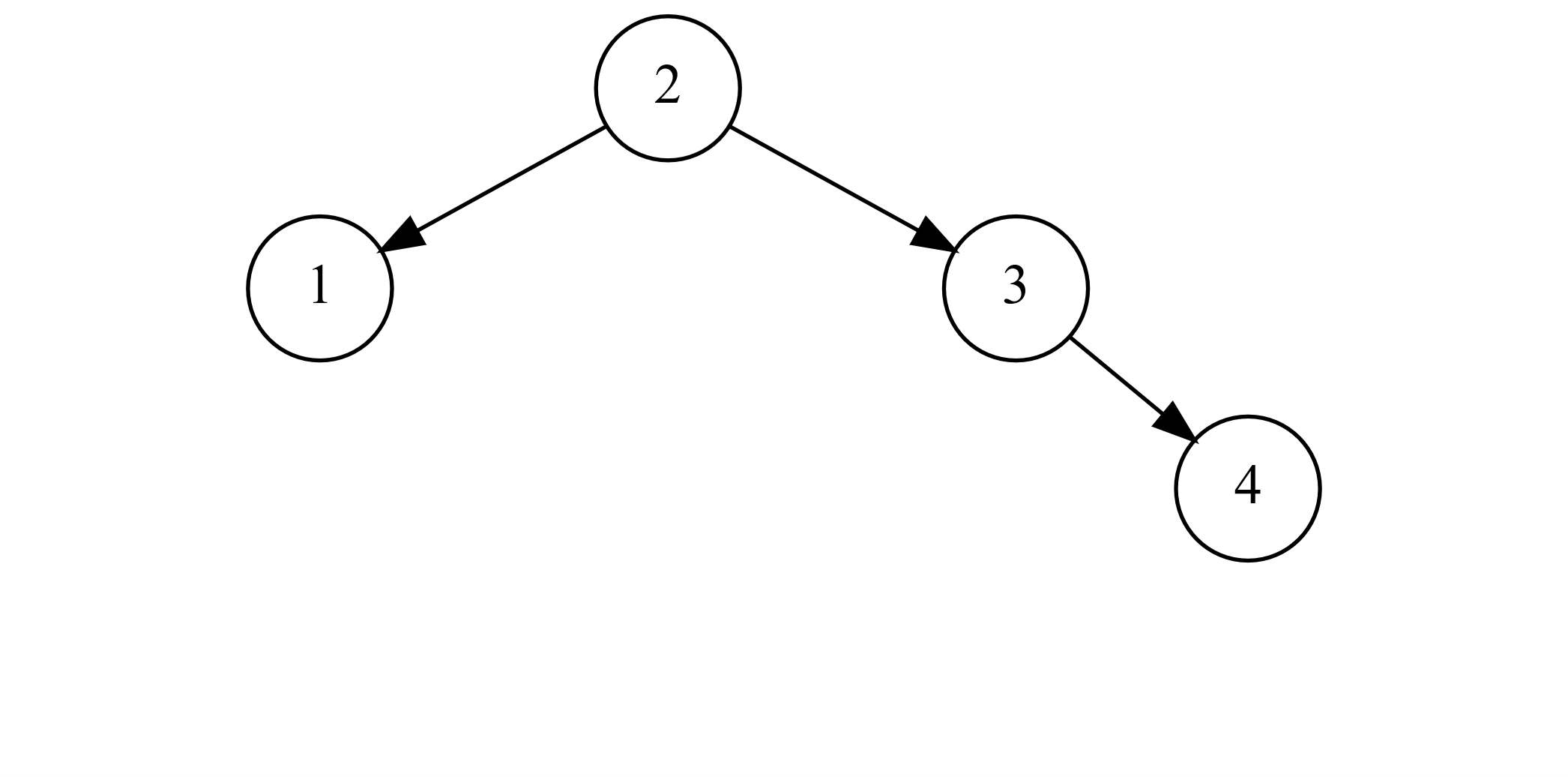
Insert 3 → Rotate Left



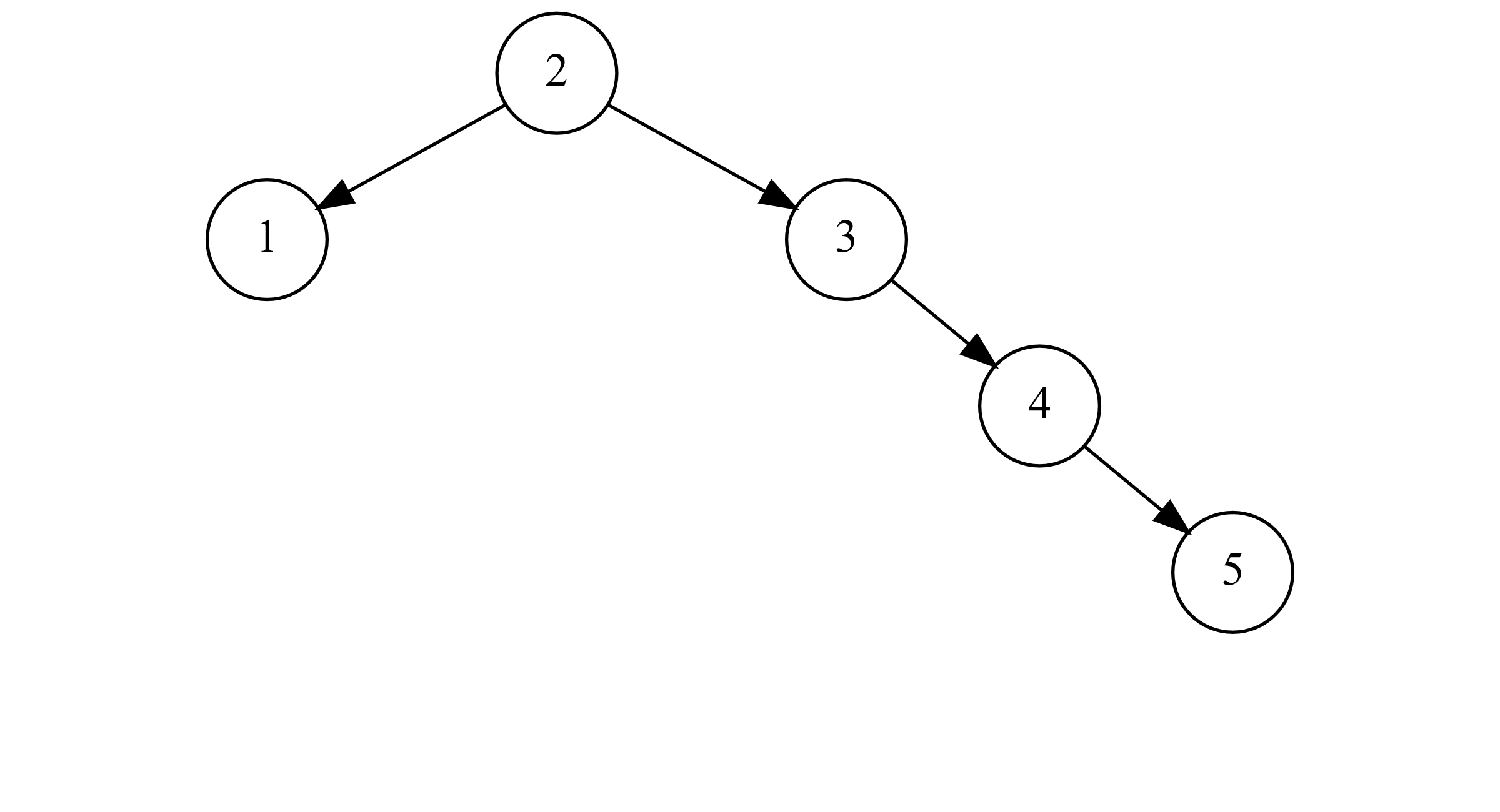
Rotate Left Result



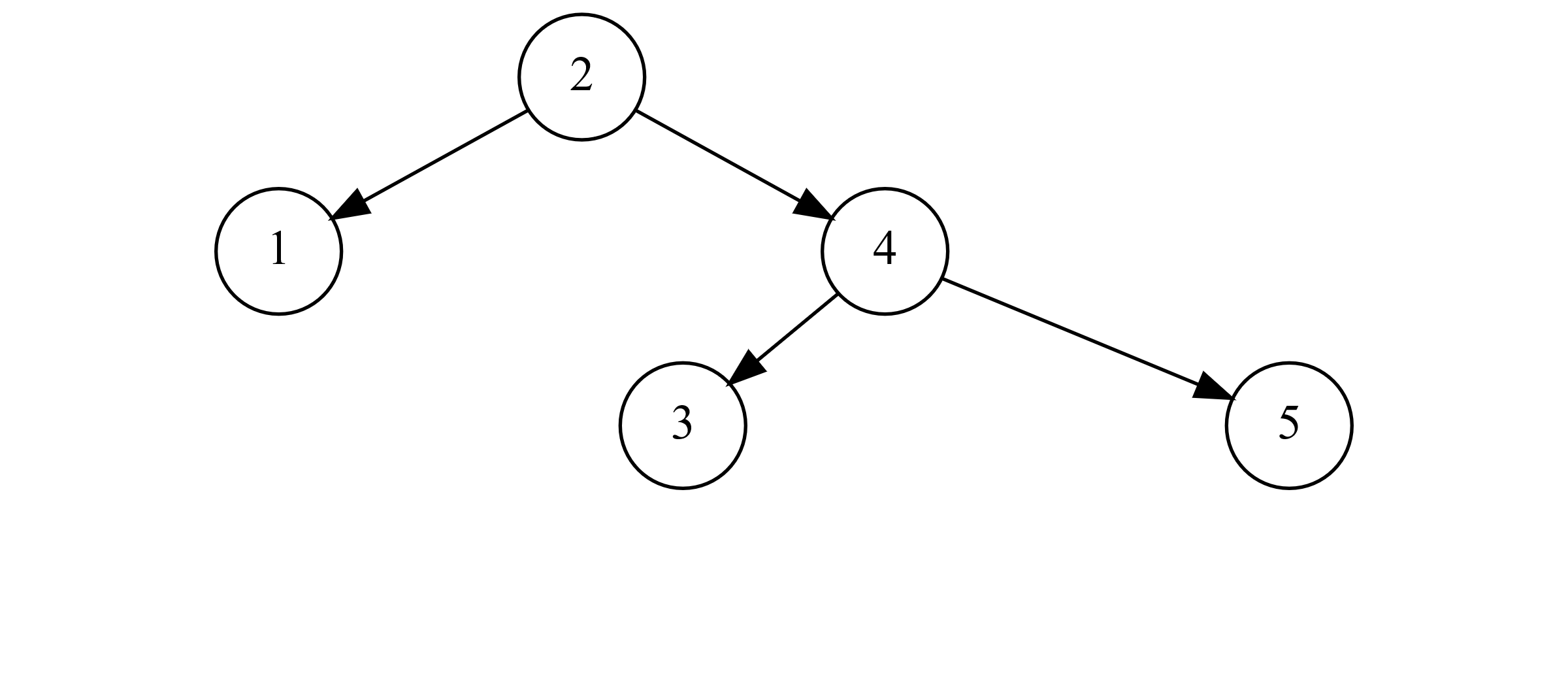
Insert 4



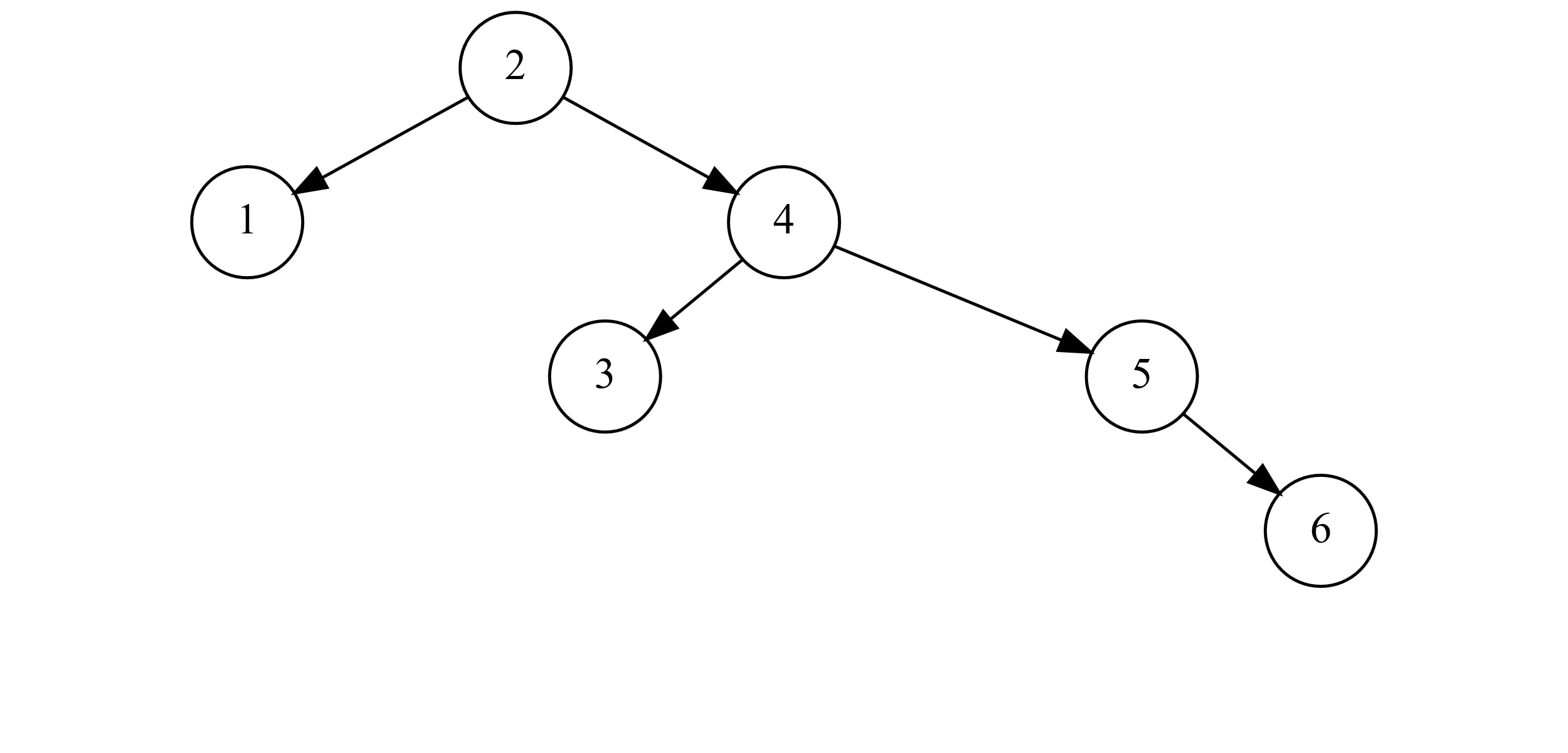
Insert 5 → Rotate Left



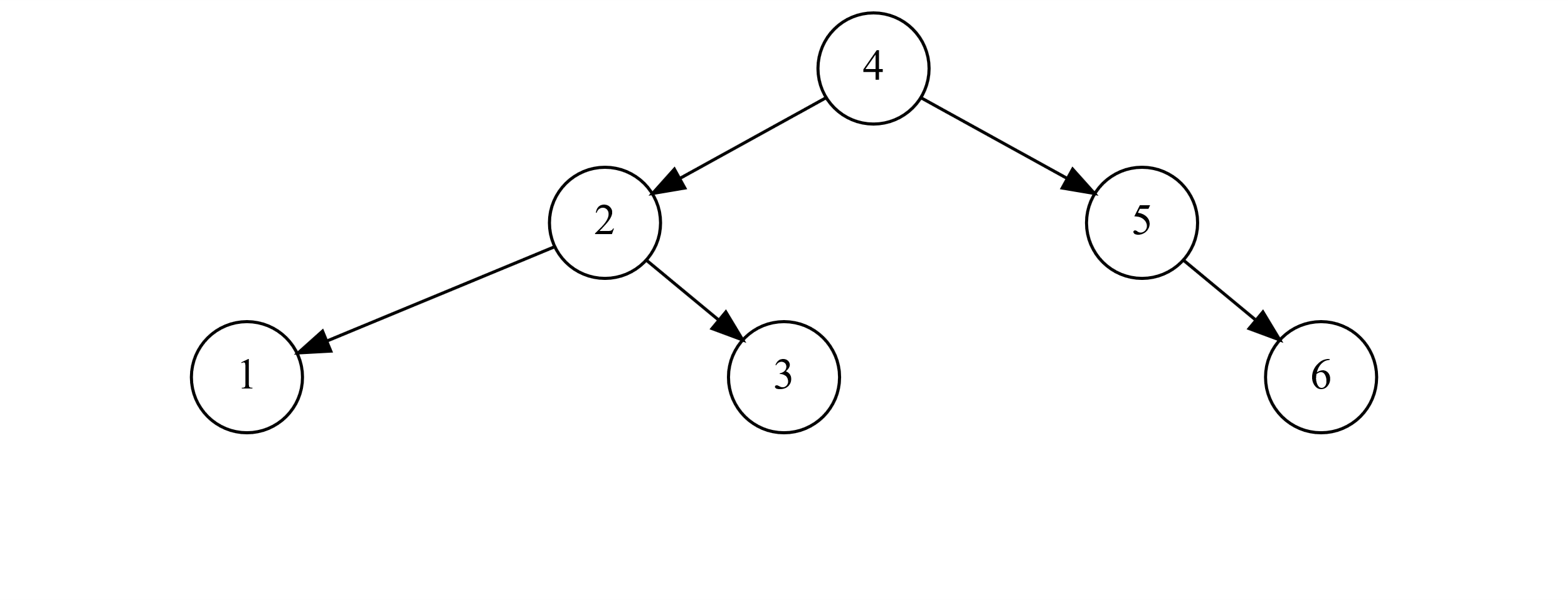
Rotate Left Result



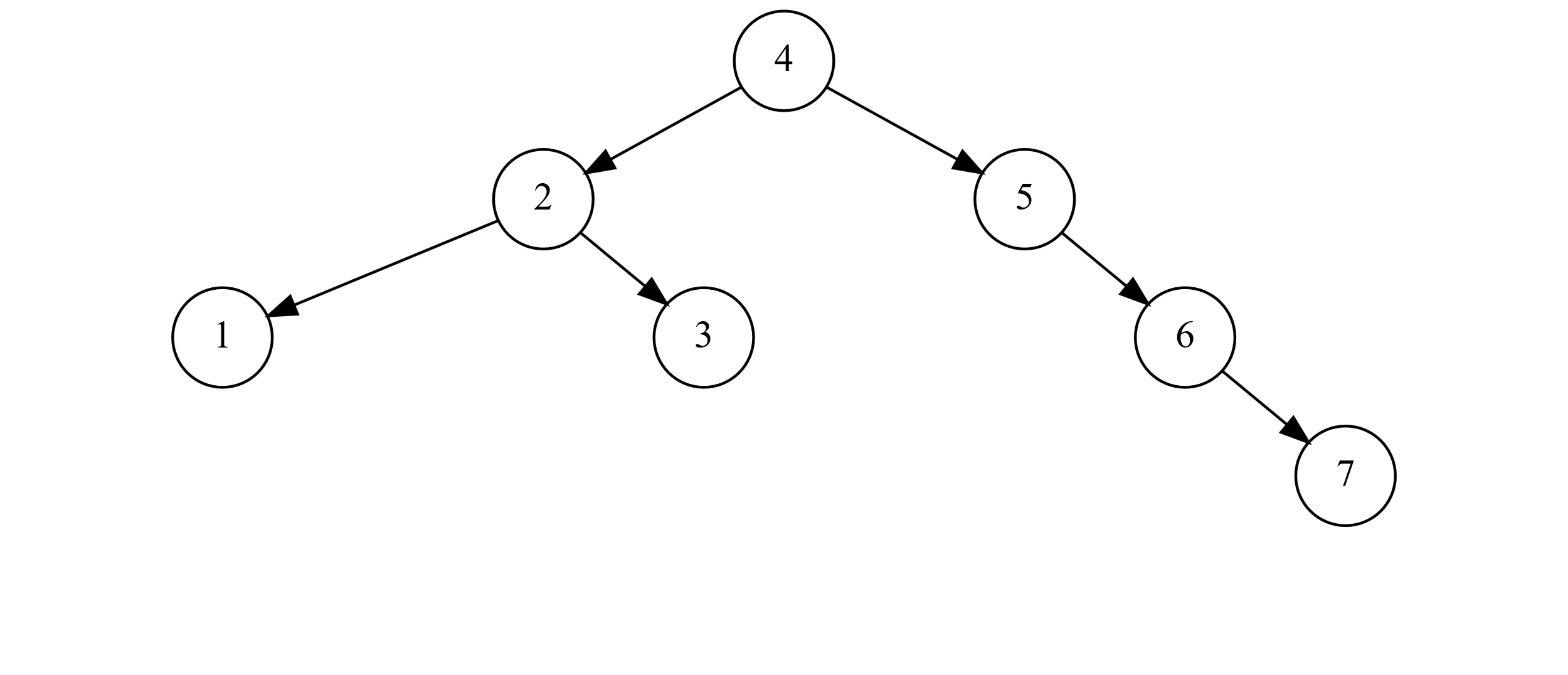
Insert 6 → Rotate Left



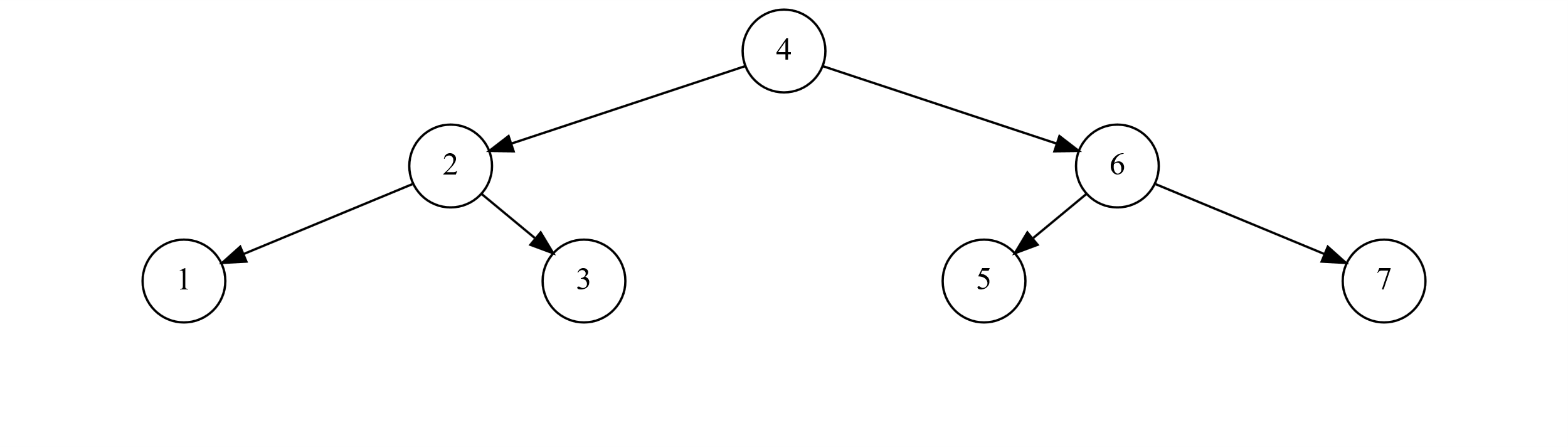
Rotate Left Result

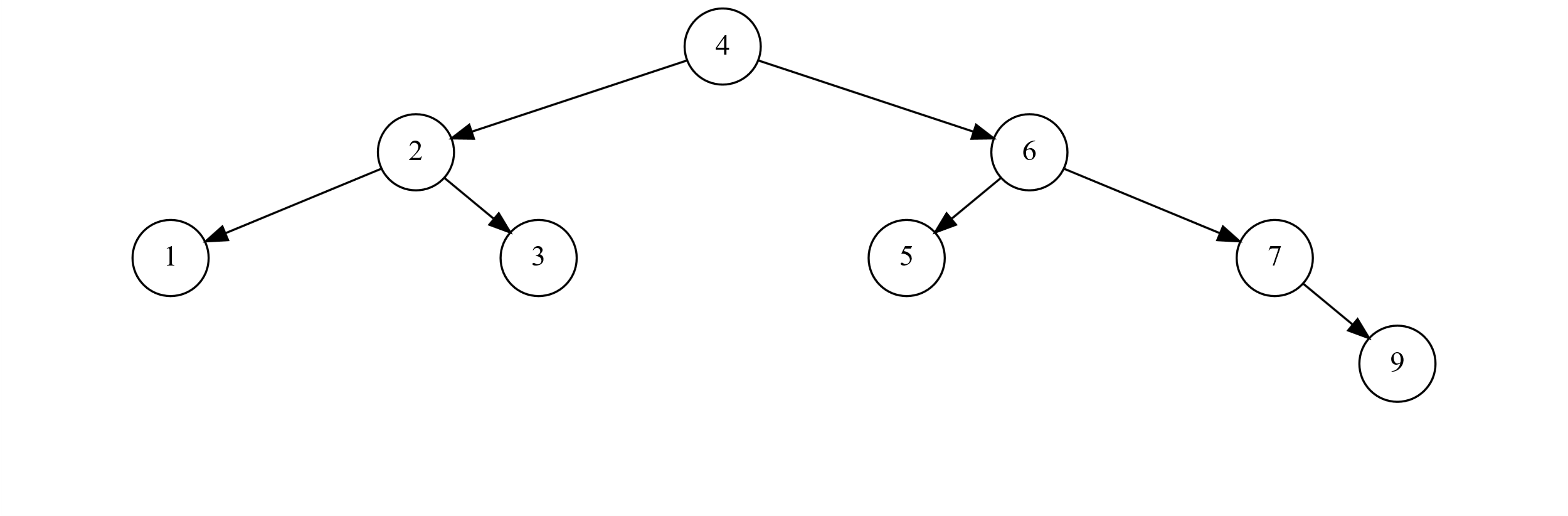


Insert 7 → Rotate Left

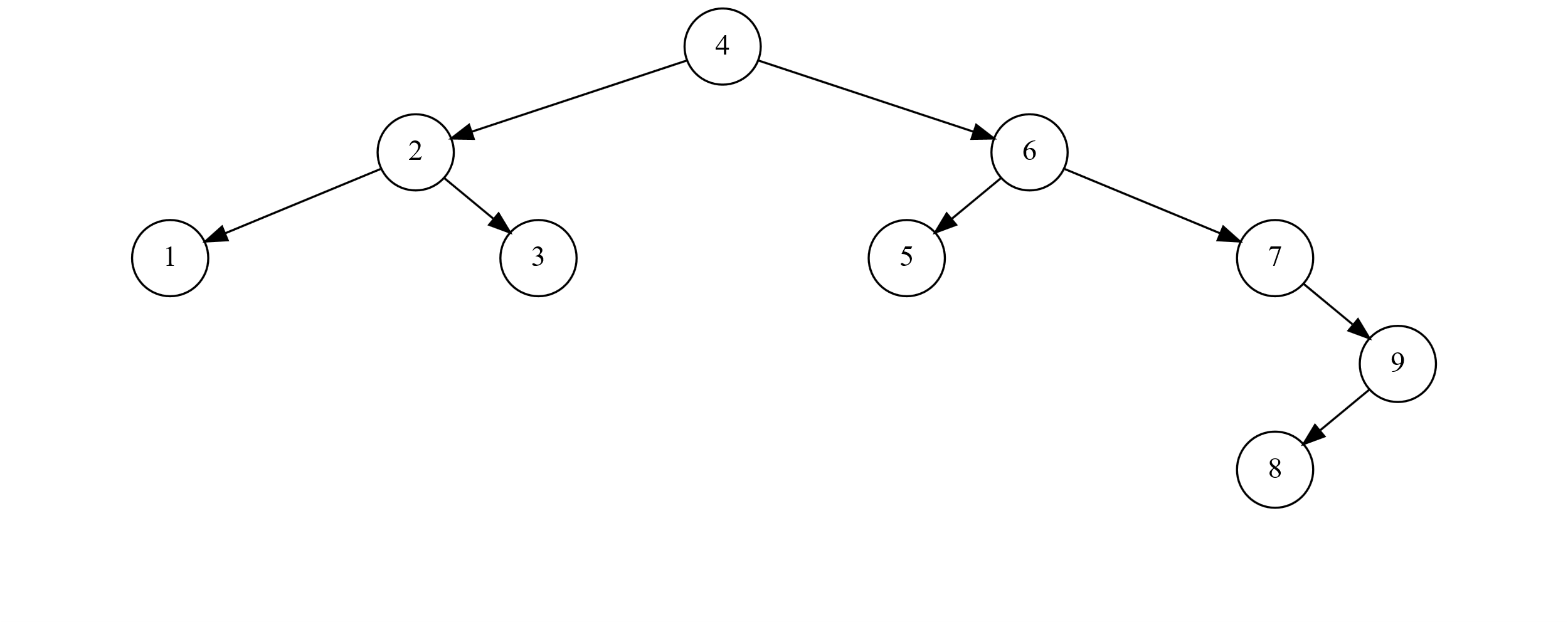


Rotate Left Result

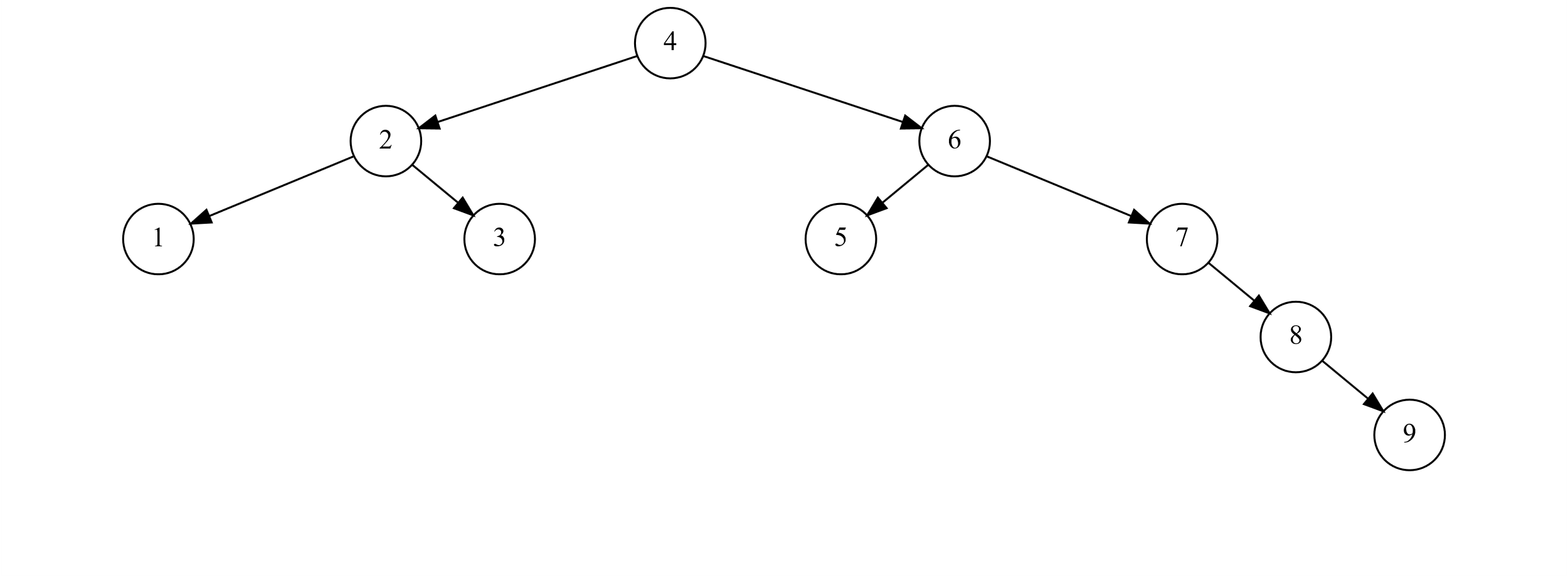


Insert 9 

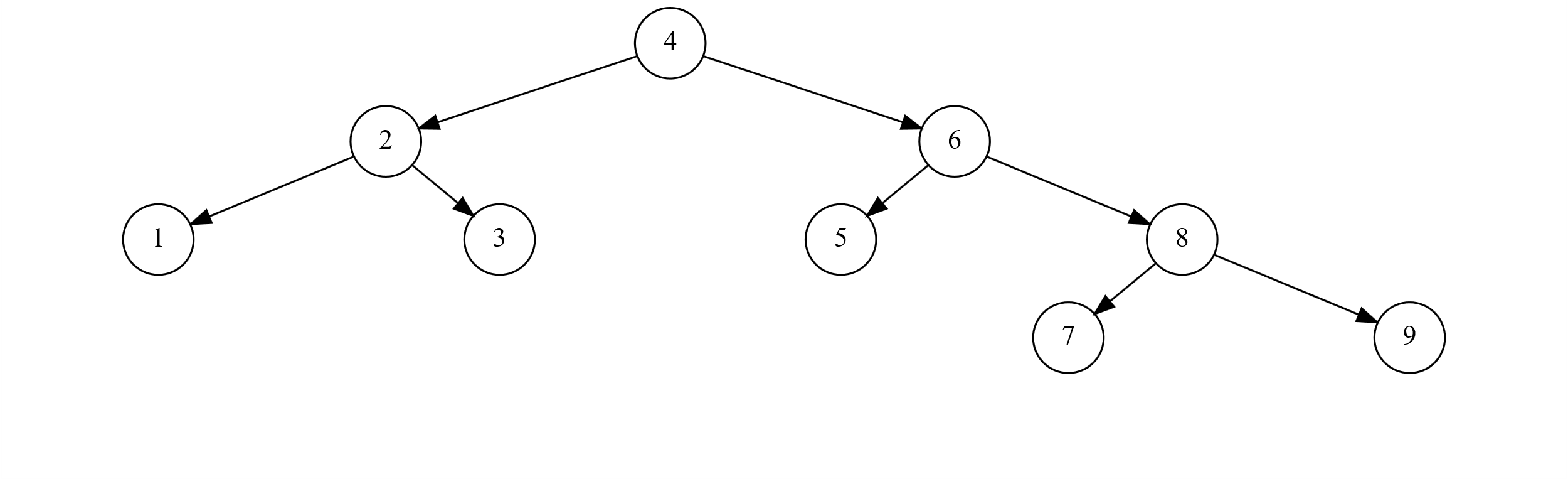
Insert 8 → (Double Rotate) Rotate Right



Rotate Right Result → Rotate Left

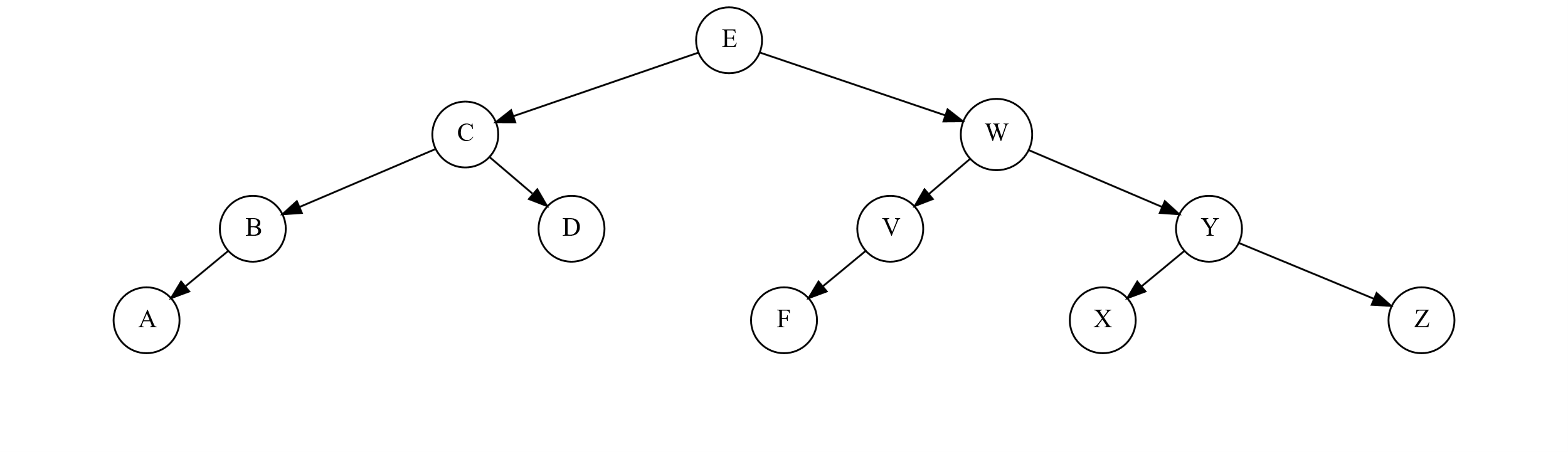


Rotate Left Result

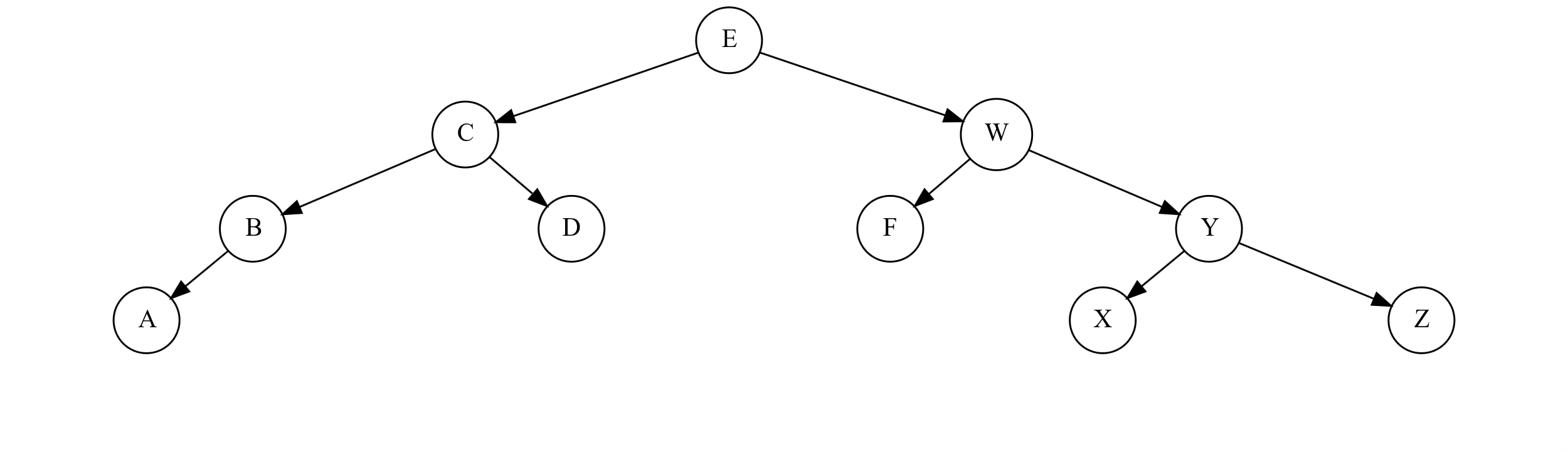


Exercise 3: Delete and Rotate

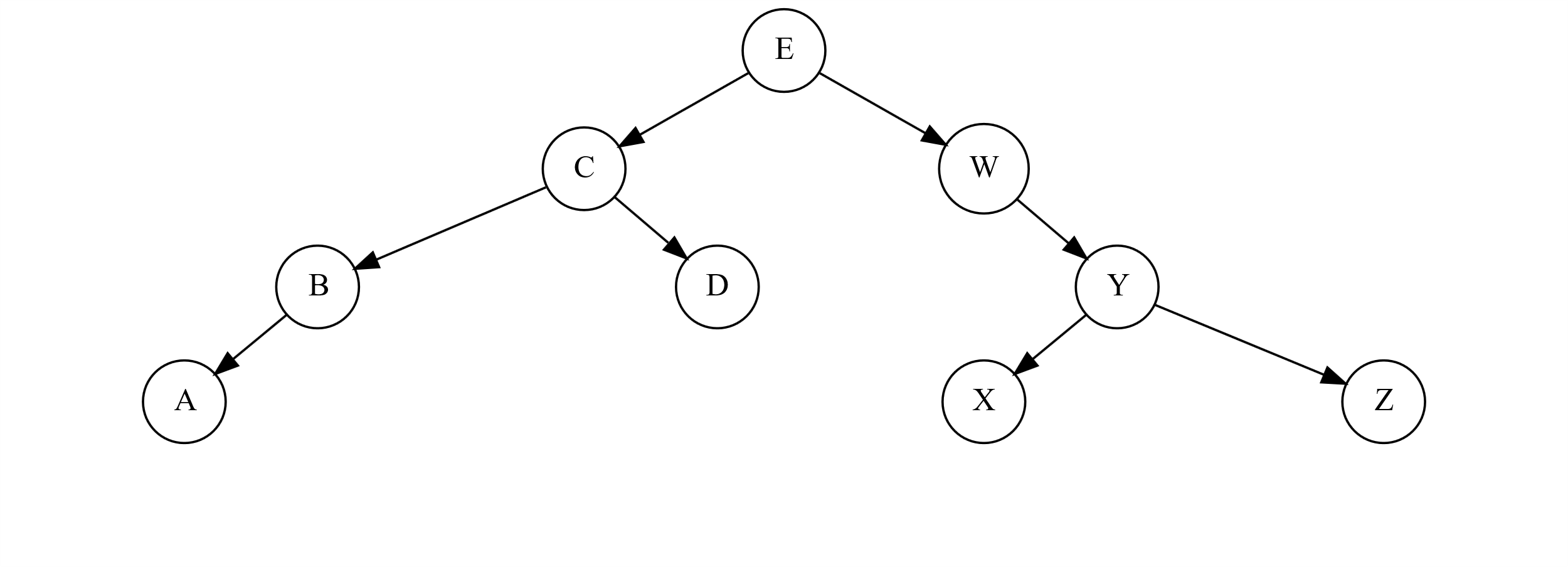
Given AVL Tree



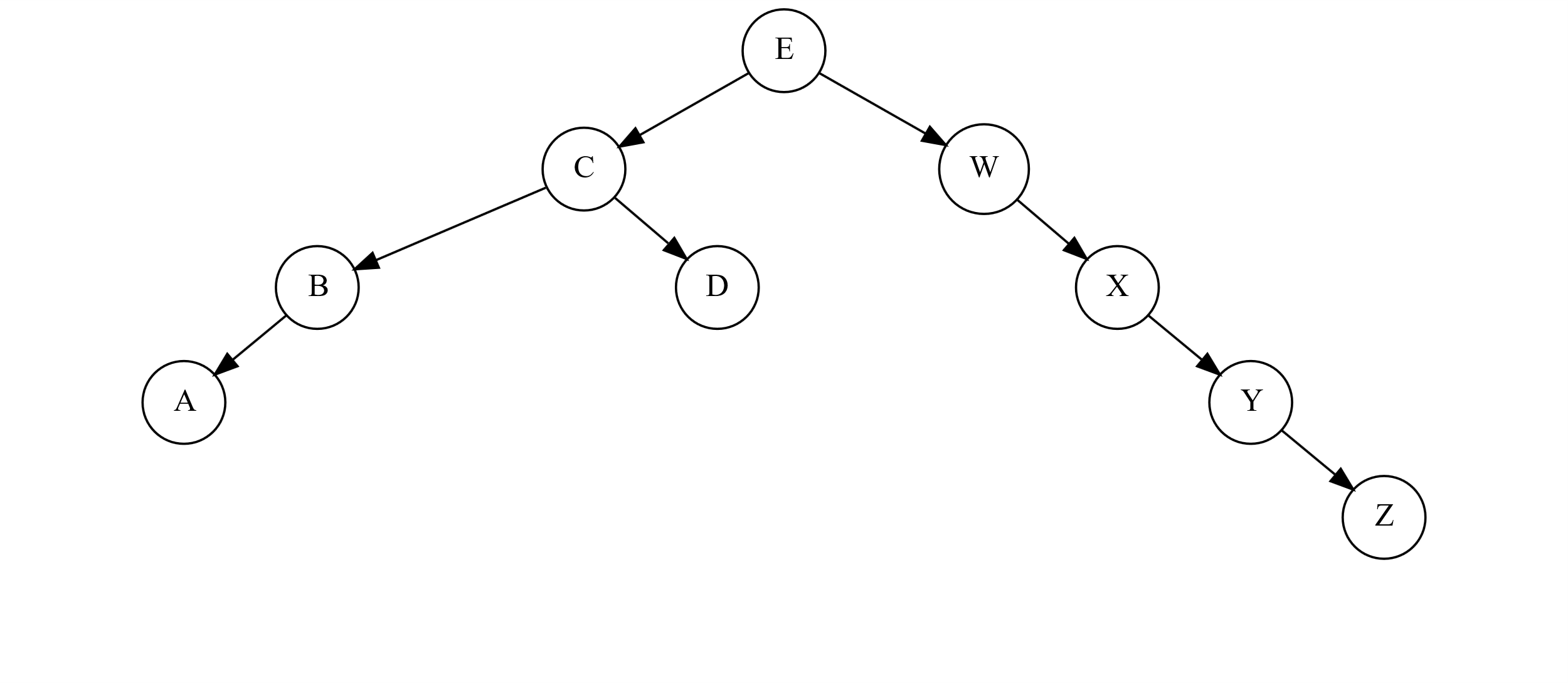
Delete V



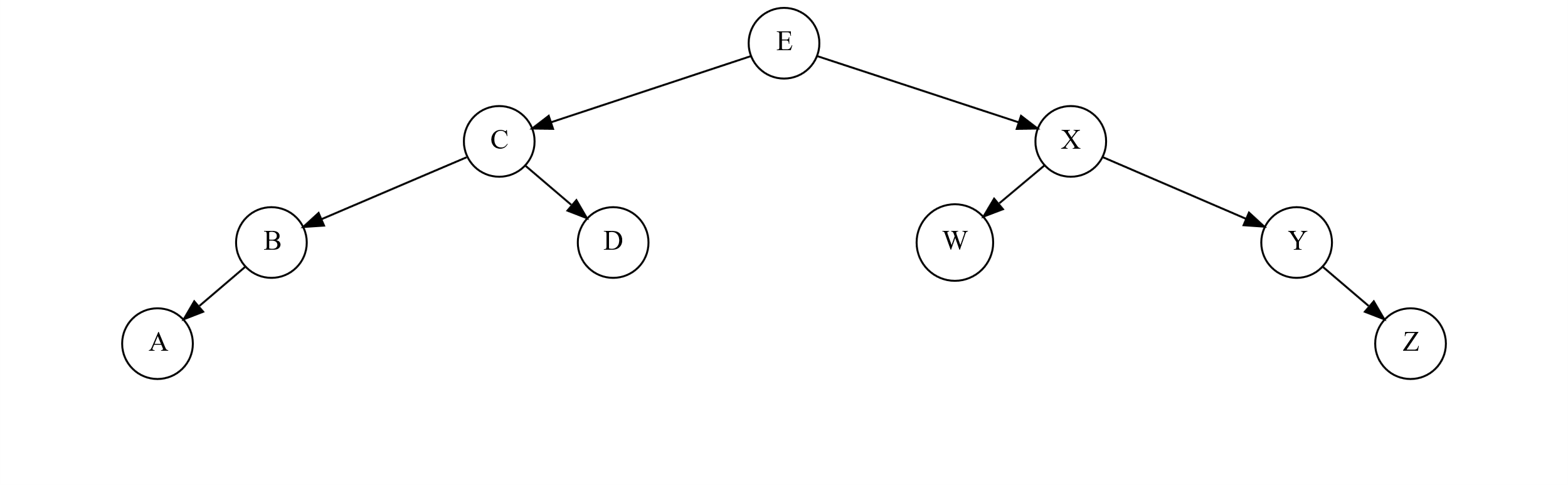
Delete F → (Double Rotate could have been one Left Rotate instead) Rotate Right



Rotate Right Result → Rotate Left



Rotate Left Result



**Assignment #3-B: Hash Functions & Hash Tables**

Given input { 66, 28, 43, 29, 44, 69, 19 } and a hash , show the resulting hash table:

1. Using Separate Chaining

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Function | Index | **Index** | **Element** |  |  |
| *h(66) =* | 6 | 0 |  |  |  |
| *h(28) =* | 8 | 1 |  |  |  |
| *h(43) =* | 3 | 2 | 43 |  |  |
| *h(29) =* | 9 | 3 | 44 |  |  |
| *h(44) =* | 4 | 4 |  |  |  |
| *h(69) =* | 9 | 5 | 66 |  |  |
| *h(19) =* | 9 | 6 |  |  |  |
|  |  | 7 |  |  |  |
|  |  | 8 | 28 |  |  |
|  |  | 9 | 19 | 69 | 29 |

1. Using Linear Probing

Where

|  |  |  |  |
| --- | --- | --- | --- |
| Function | Index | **Index** | **Element** |
| *h(66 + f(0)) =* | 6 | 0 | 69 |
| *h(28 + f(0)) =* | 8 | 1 | 19 |
| *h(43 + f(0)) =* | 3 | 2 |  |
| *h(29 + f(0)) =* | 9 | 3 | 43 |
| *h(44 + f(0)) =* | 4 | 4 | 44 |
| *h(69 + f(0)) =* | 9 | 5 |  |
| *h(69 + f(1)) =* | 0 | 6 | 66 |
| *h(19 + f(0)) =* | 9 | 7 |  |
| *h(19 + f(1)) =* | 0 | 8 | 28 |
| *h(19 + f(2)) =* | 1 | 9 | 29 |

1. Using Quadratic Probing

Where

|  |  |  |  |
| --- | --- | --- | --- |
| Function | Index | **Index** | **Element** |
| *h(66 + f(0)) =* | 6 | 0 | 69 |
| *h(28 + f(0)) =* | 8 | 1 |  |
| *h(43 + f(0)) =* | 3 | 2 |  |
| *h(29 + f(0)) =* | 9 | 3 | 43 |
| *h(44 + f(0)) =* | 4 | 4 | 44 |
| *h(69 + f(0)) =* | 9 | 5 | 19 |
| *h(69 + f(1)) =* | 0 | 6 | 66 |
| *h(19 + f(0)) =* | 9 | 7 |  |
| *h(19 + f(1)) =* | 0 | 8 | 28 |
| *h(19 + f(2)) =* | 3 | 9 | 29 |
| *h(19 + f(3)) =* | 8 |  |  |
| *h(19 + f(3)) =* | 5 |  |  |

1. Starting with the following hash function: apply Rehashing as described in the primary course slides.

|  |  |  |  |
| --- | --- | --- | --- |
| Function | Index | **Index** | **Element** |
| *h(66) =* | 15 | 0 |  |
| *h(28) =* | 11 | 1 | 69 |
| *h(43) =* | 9 | 2 | 19 |
| *h(29) =* | 12 | 3 |  |
| *h(44) =* | 10 | 4 |  |
| *h(69) =* | 1 | 5 |  |
| *h(19) =* | 2 | 6 |  |
|  |  | 7 |  |
|  |  | 8 |  |
|  |  | 9 | 43 |
|  |  | 10 | 44 |
|  |  | 11 | 28 |
|  |  | 12 | 29 |
|  |  | 13 |  |
|  |  | 14 |  |
|  |  | 15 | 66 |
|  |  | 16 |  |
|  |  | 17 |  |

Since load factor is

The equation must be rehashed using the next prime number that is greater than the new table size where the table size is doubled to 14.

If we are changing the hashing function, why not remove the portion to not deal with negatives due to the range of mod 17 being from 0 to 16. Why make a hashing function unnecessarily complicated?