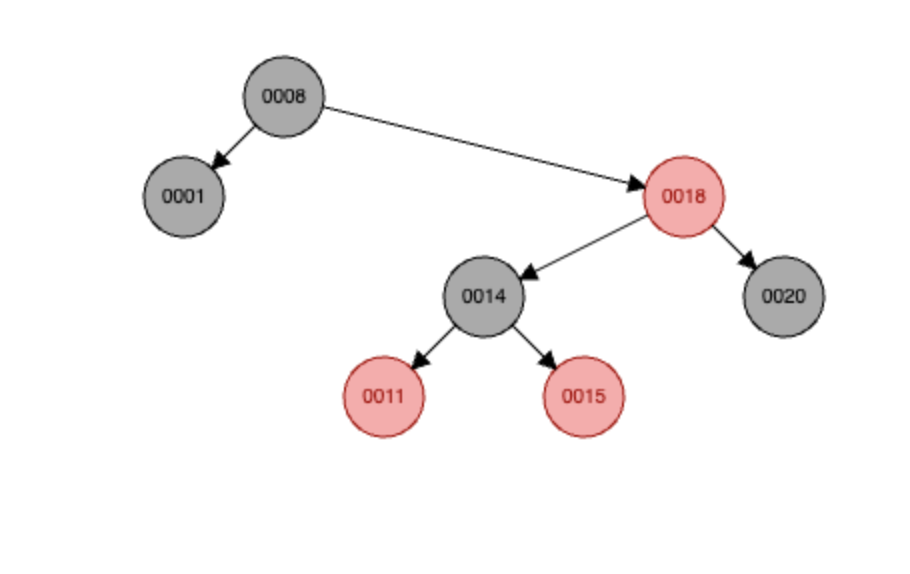
# Answer of 1



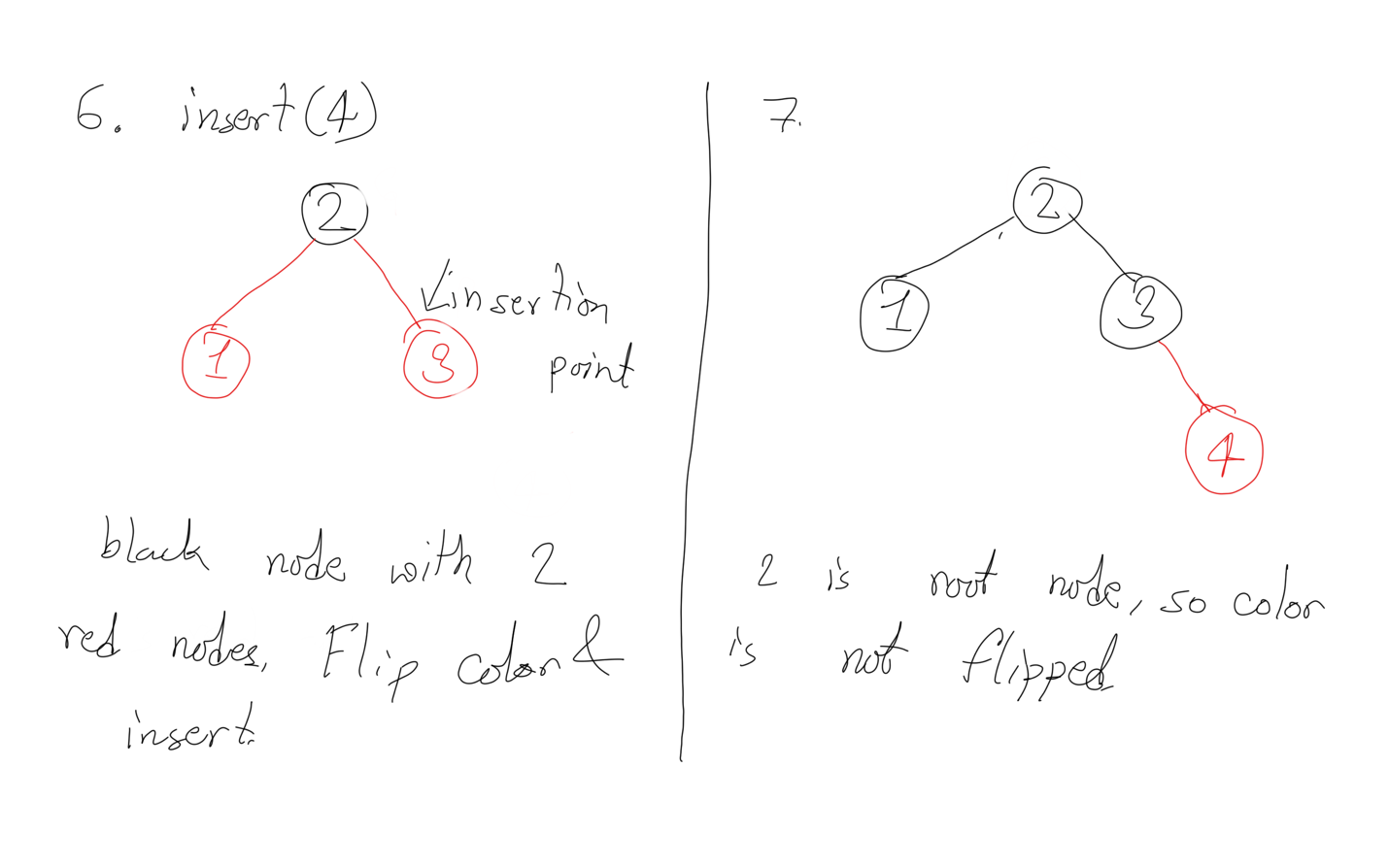
# Answer of 2

1. Insert [1,2,3,4,5,6,7,8]

A group of red circles with numbers on a black background with Marfa lights in the background

Description automatically generatedA black background with red circles

Description automatically generated

A black background with red text

Description automatically generatedA group of numbers on a black background with Marfa lights in the background

Description automatically generatedA black background with red circles and numbers

Description automatically generatedA black background with red circles and numbers

Description automatically generated A black screen with red circles and numbers

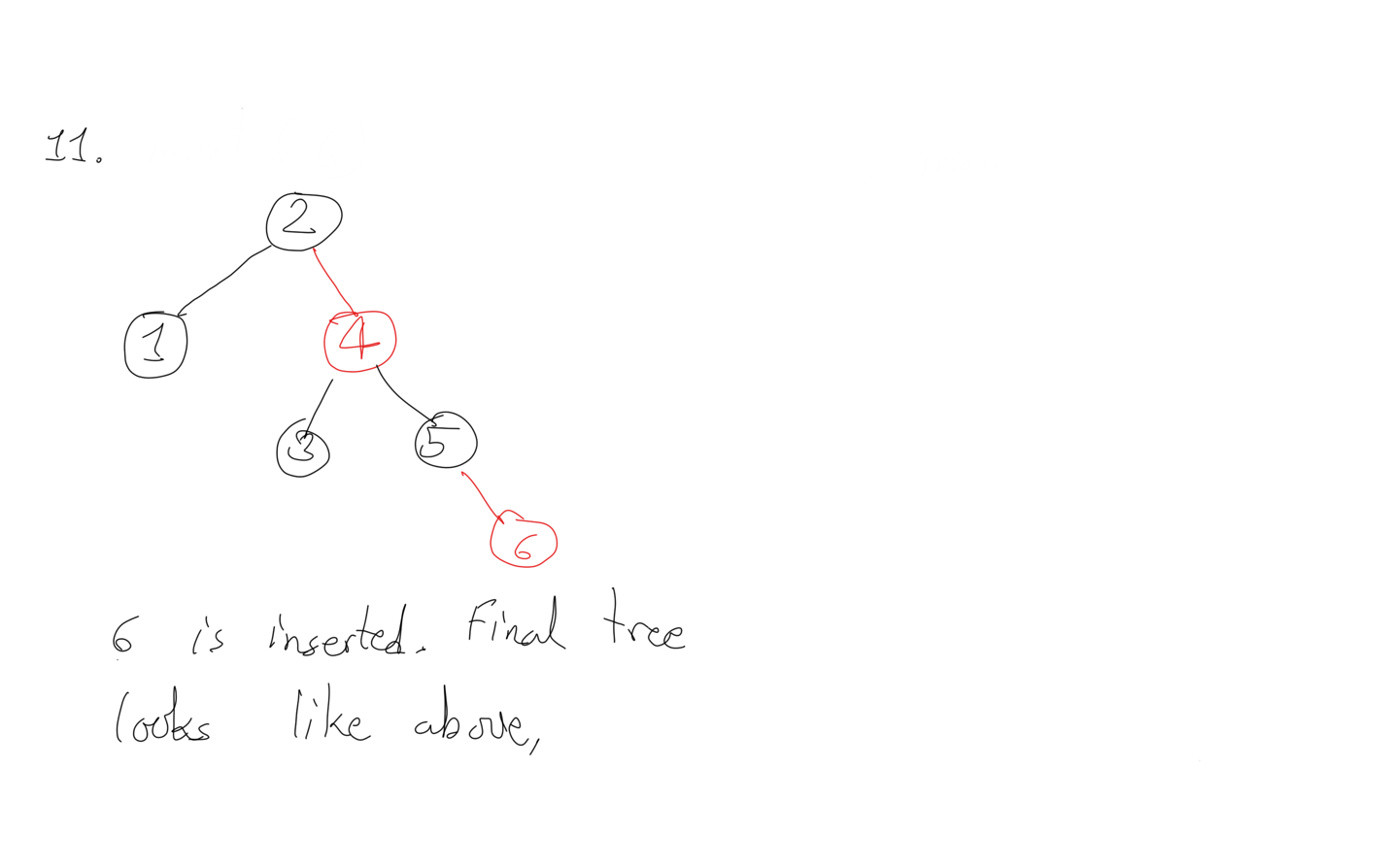
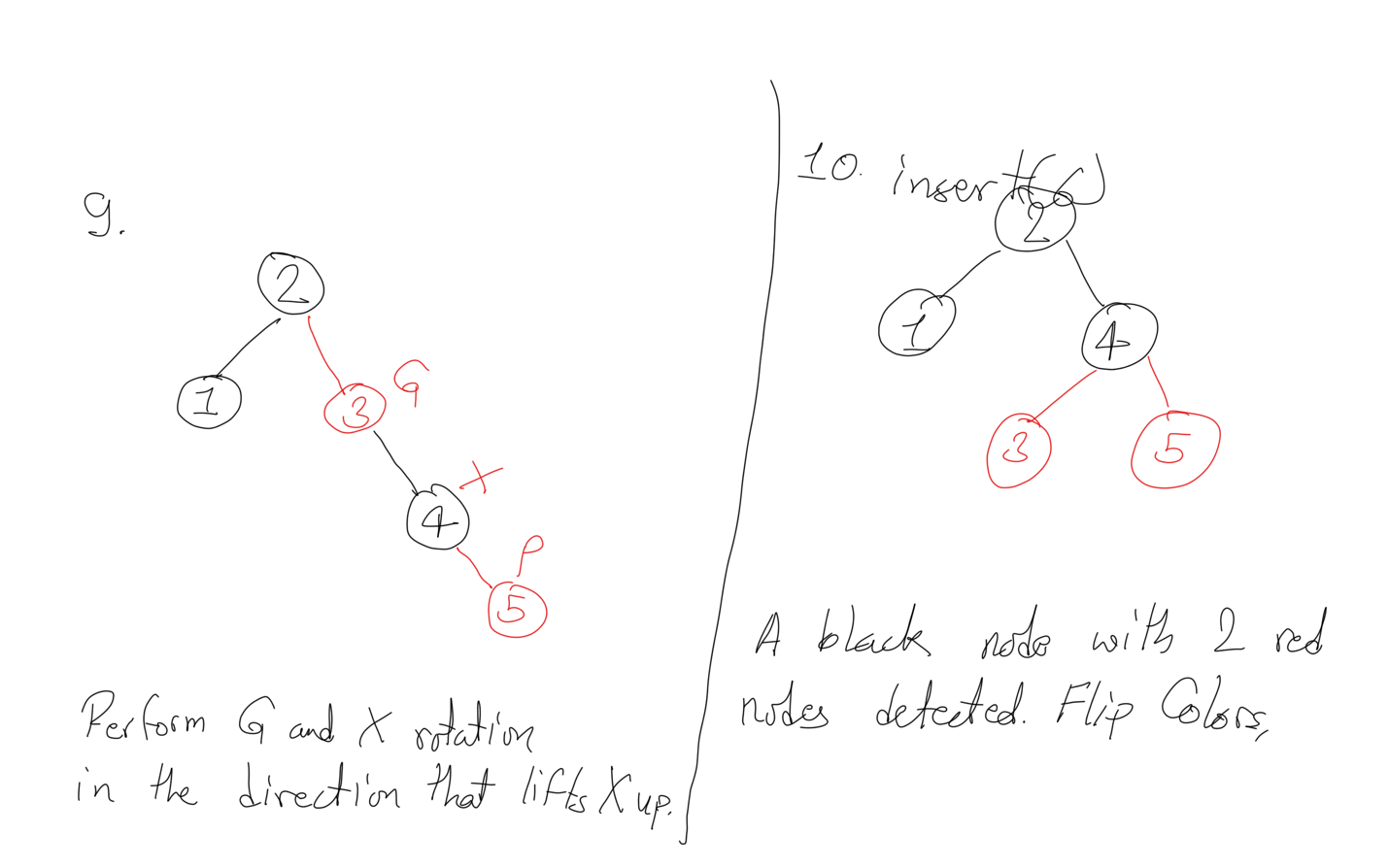
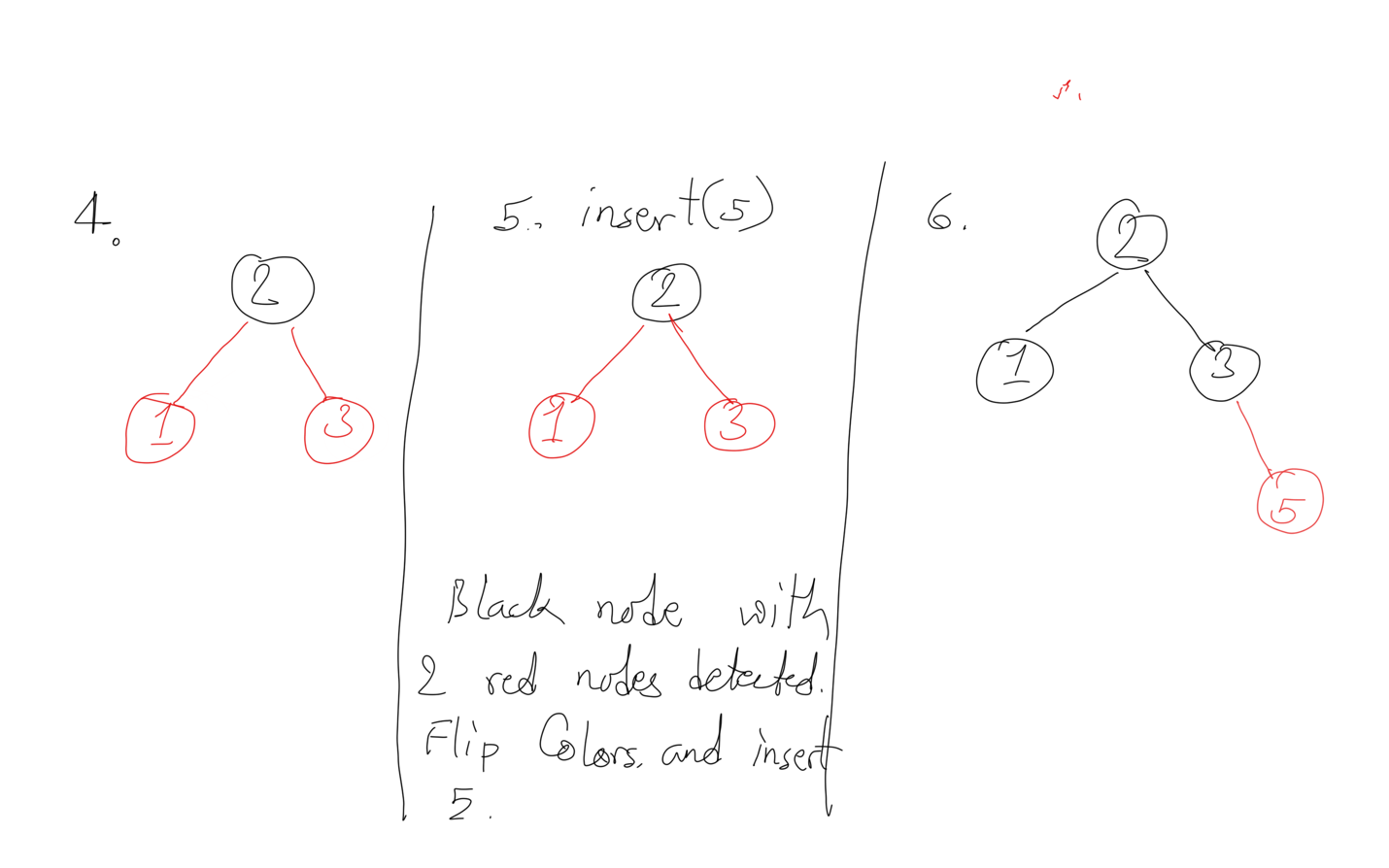
Description automatically generatedA black background with red lines

Description automatically generated

1. [3,2,1 5,4,6]

A black background with red circles and numbers with Marfa lights in the background

Description automatically generated



# Answer of 3

Hashmap Implementation:

class Trie {

static class TrieNode {

public Map<Character, TrieNode> children;

public boolean isEndOfWord;

TrieNode() {

children = new HashMap<>();

isEndOfWord = false;

}

}

TrieNode root;

public Trie() {

root = new TrieNode();

}

public void insert(String word) {

TrieNode current = root;

for (var c : word.toCharArray()) {

current = current.children.computeIfAbsent(c,k -> new TrieNode());

}

current.isEndOfWord = true;

}

public boolean search(String word) {

TrieNode current = root;

for (var c : word.toCharArray()) {

current = current.children.getOrDefault(c, null);

if (current == null) {

return false;

}

}

return current.isEndOfWord;

}

public boolean startsWith(String prefix) {

TrieNode current = root;

for (var c : prefix.toCharArray()) {

current = current.children.getOrDefault(c, null);

if (current == null) {

return false;

}

}

return true;

}

}

/\*\*

\* Your Trie object will be instantiated and called as such:

\* Trie obj = new Trie();

\* obj.insert(word);

\* boolean param\_2 = obj.search(word);

\* boolean param\_3 = obj.startsWith(prefix);

\*/

Array Implementation:

class Trie {

static class TrieNode {

public TrieNode[] children;

public boolean isEndOfWord;

TrieNode() {

children = new TrieNode[26];

isEndOfWord = false;

}

}

TrieNode root;

public Trie() {

root = new TrieNode();

}

public void insert(String word) {

TrieNode current = root;

for (var c : word.toCharArray()) {

int index = c - 'a';

if (current.children[index] == null){

current.children[index] = new TrieNode();

}

current = current.children[index];

}

current.isEndOfWord = true;

}

public boolean search(String word) {

TrieNode current = root;

for (var c : word.toCharArray()) {

int index = c - 'a';

current = current.children[index];

if (current == null) {

return false;

}

}

return current.isEndOfWord;

}

public boolean startsWith(String prefix) {

TrieNode current = root;

for (var c : prefix.toCharArray()) {

int index = c - 'a';

current = current.children[index];

if (current == null) {

return false;

}

}

return true;

}

}

/\*\*

\* Your Trie object will be instantiated and called as such:

\* Trie obj = new Trie();

\* obj.insert(word);

\* boolean param\_2 = obj.search(word);

\* boolean param\_3 = obj.startsWith(prefix);

\*/