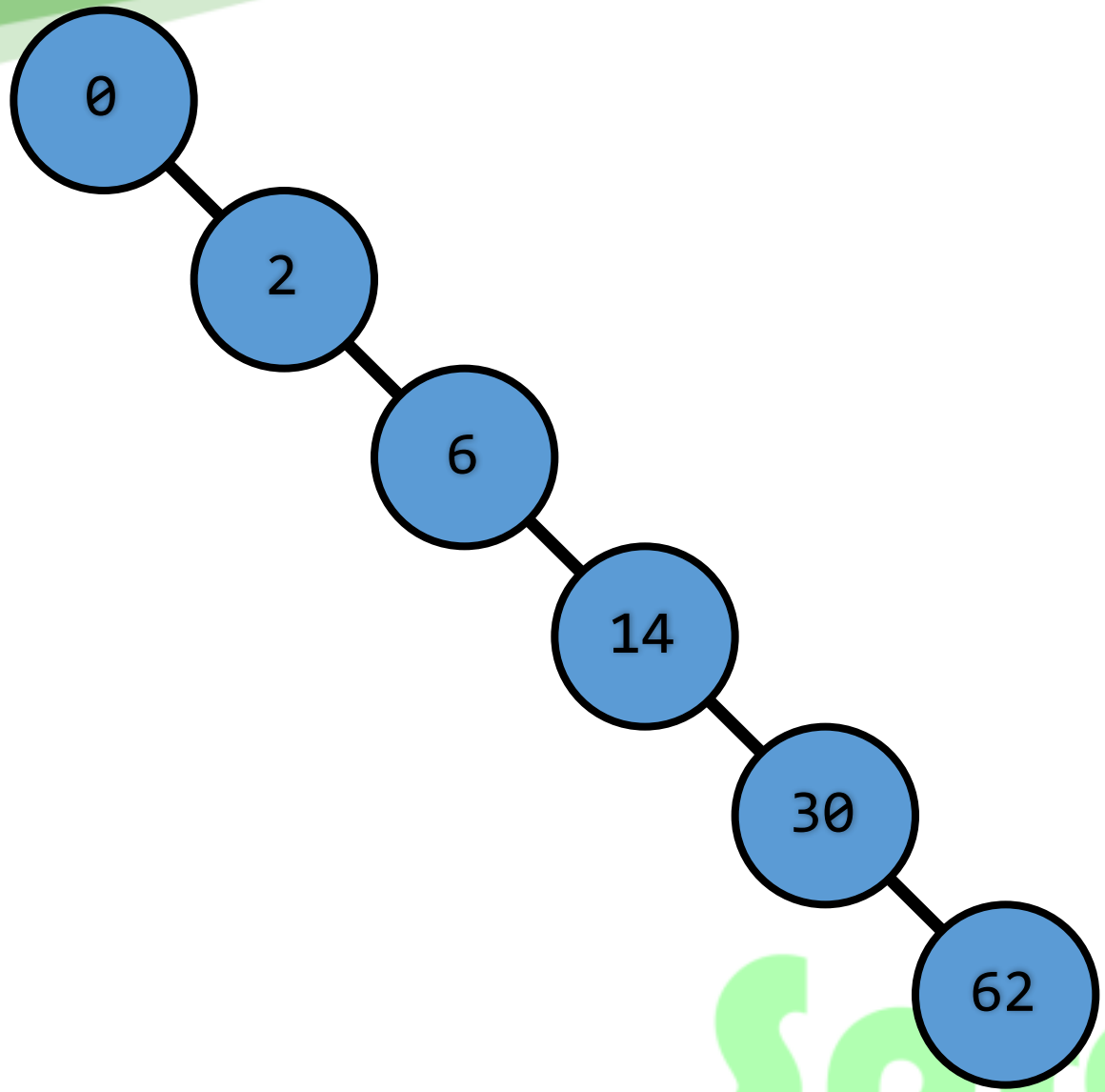
Tree
課堂補充

by zolution
Credit by nkhg
2018/03/10

Sprout

課程影片

- 想一想，為什麼一般的binary tree不適合用陣列存呢？


0
2
6
14
30
62

vector

```
vec[0]=1
vec[1]=2
vec[2]=3
vec.back()=3
vec[0]=1
vec[1]=2
0
```

```cpp
#include <cstdio>
#include <vector>
int main()
{
    std::vector<int> vec;

    vec.push_back(1);
    vec.push_back(2);
    vec.push_back(3);

    for(int i=0;i<vec.size();i++)
        printf("vec[%d]=%d\n",i,vec[i]);

    printf("vec.back()=%d\n",vec.back());

    vec.pop_back();

    for(int i=0;i<vec.size();i++)
        printf("vec[%d]=%d\n",i,vec[i]);

    vec.clear();
    printf("%d\n",vec.size());
    return 0;
}
```

時間複雜度

- 剛剛用到的一切操作都是O(1)
- 內部的實作方法將來手寫作業會介紹

二元搜尋樹

Binary Search Tree

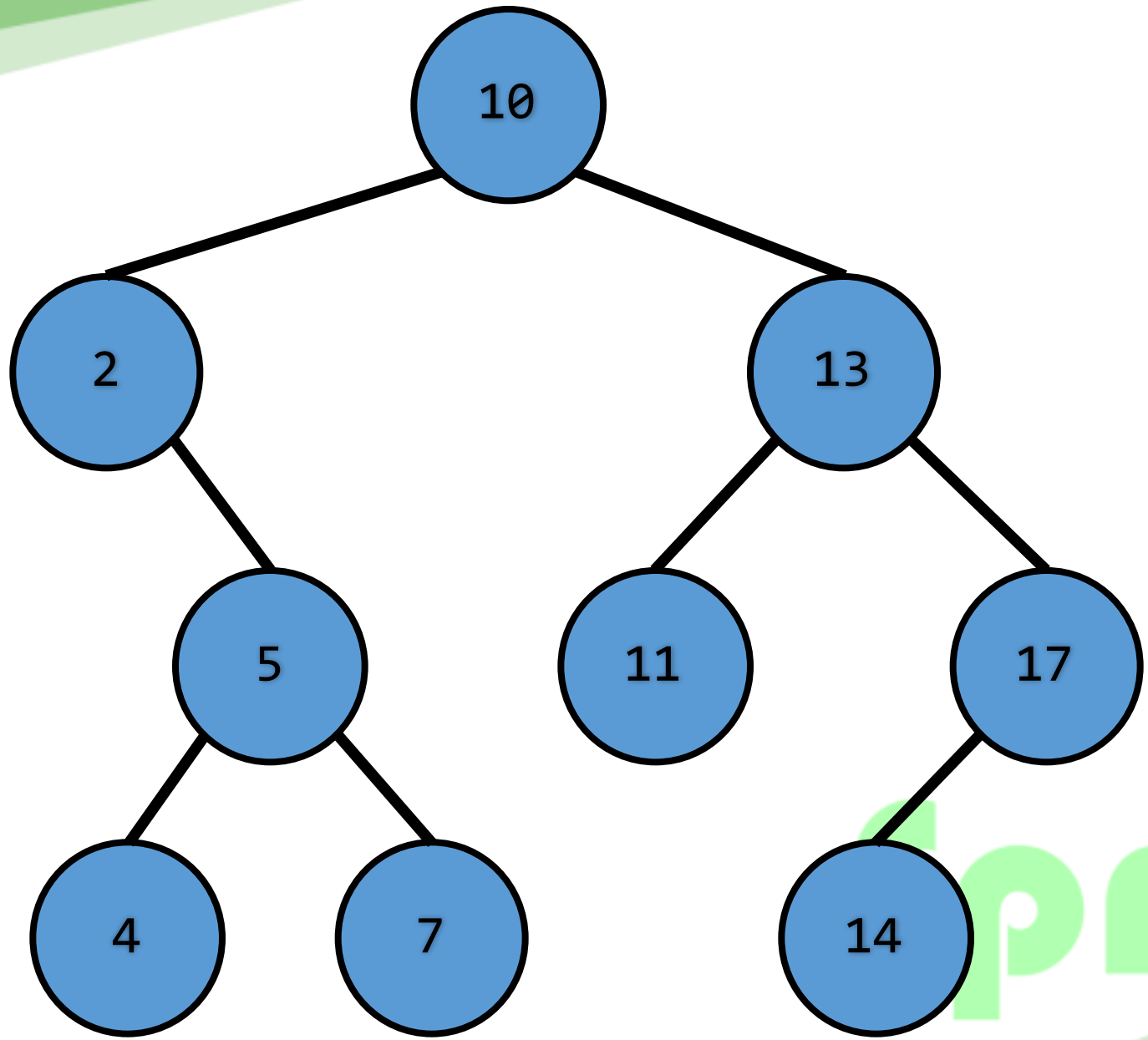
What is a binary search tree?

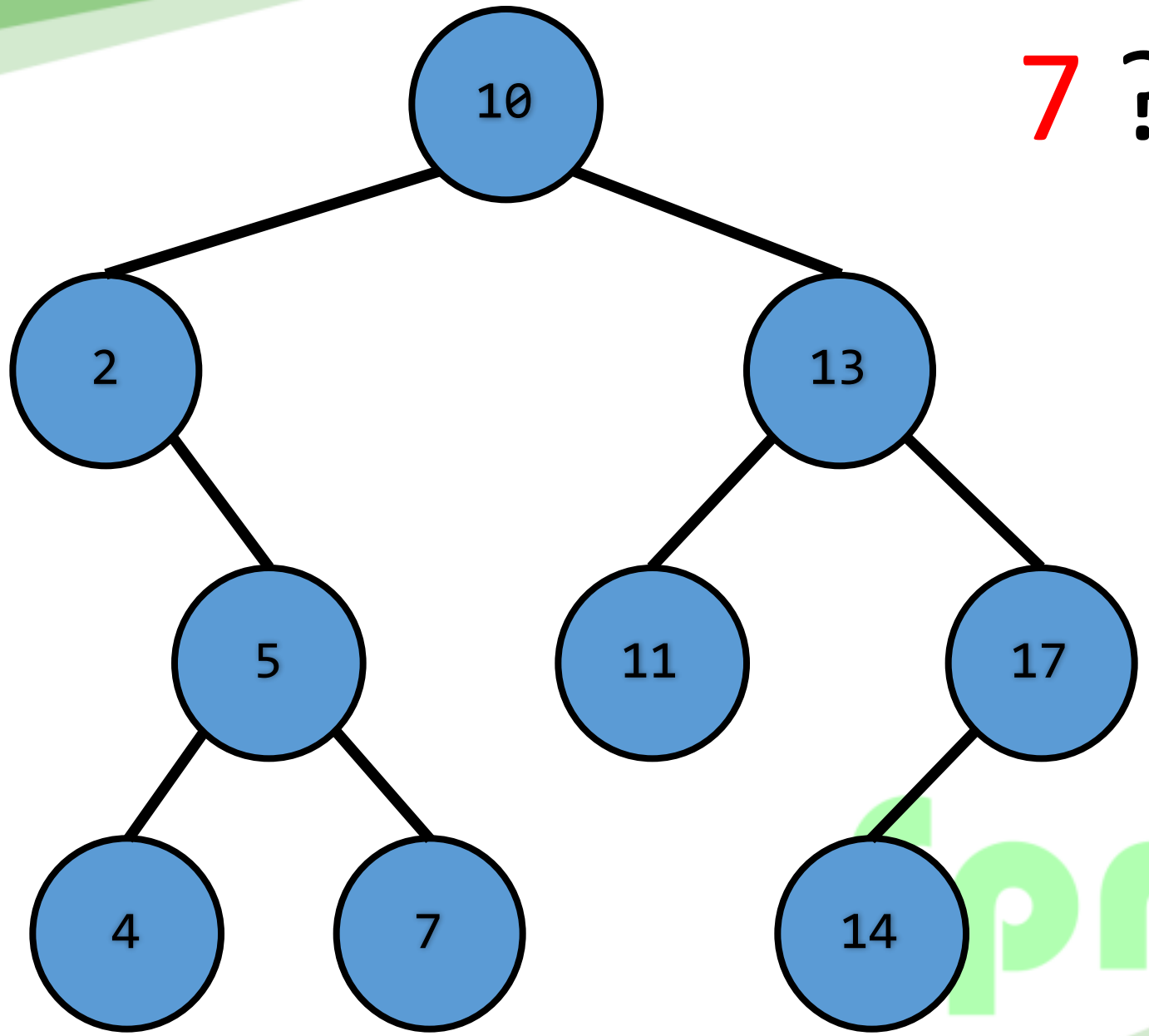
指一棵空樹或者具有下列性質的二元樹：

1. 若任意節點的左子樹不空，則左子樹上所有節點的值均小於它的根節點的值。
2. 任意節點的右子樹不空，則右子樹上所有節點的值均大於它的根節點的值。
3. 任意節點的左、右子樹也分別為二元搜尋樹。
4. 沒有鍵值相等的節點（no duplicate nodes）。

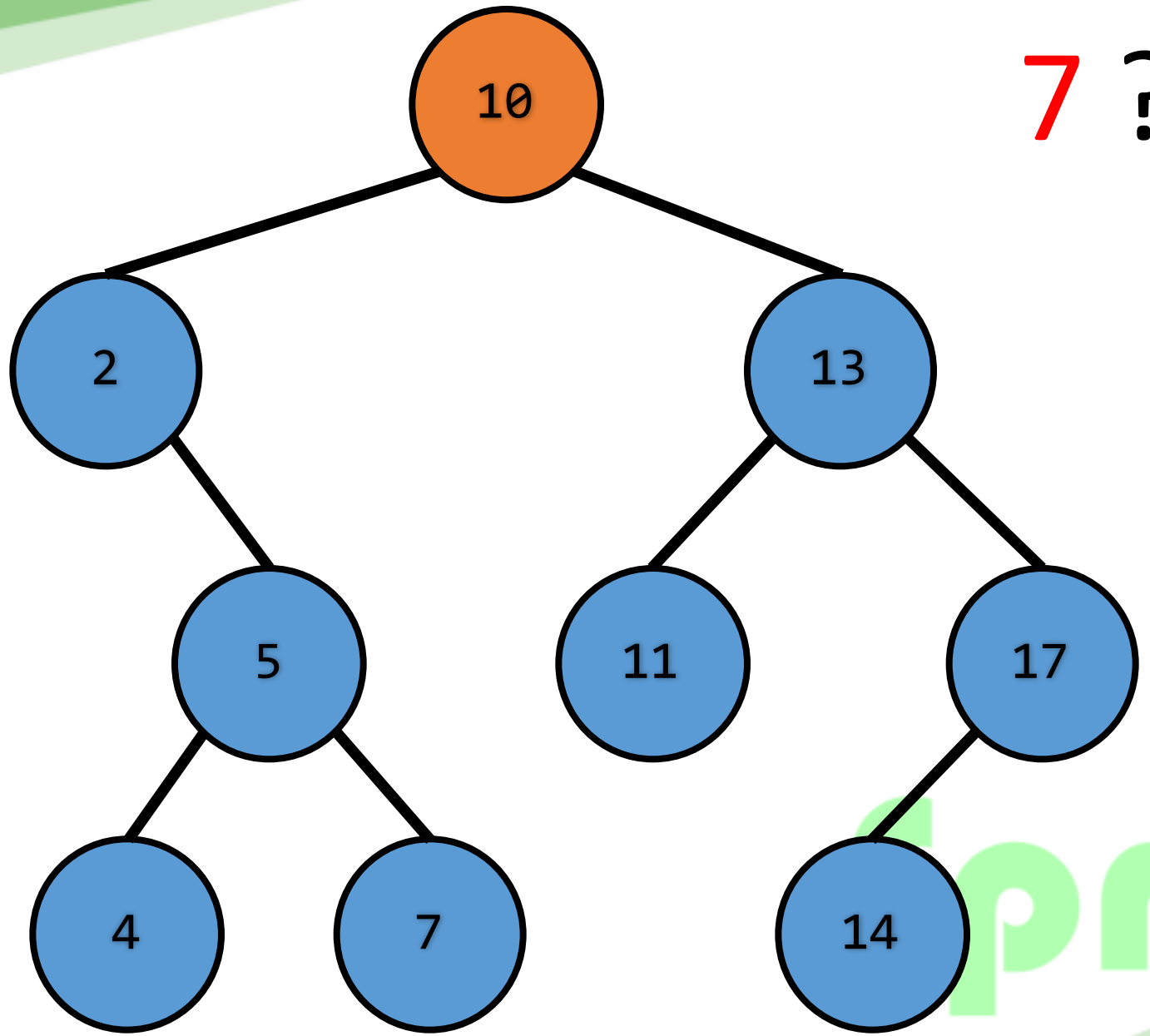
(from wikipedia)

Sprout

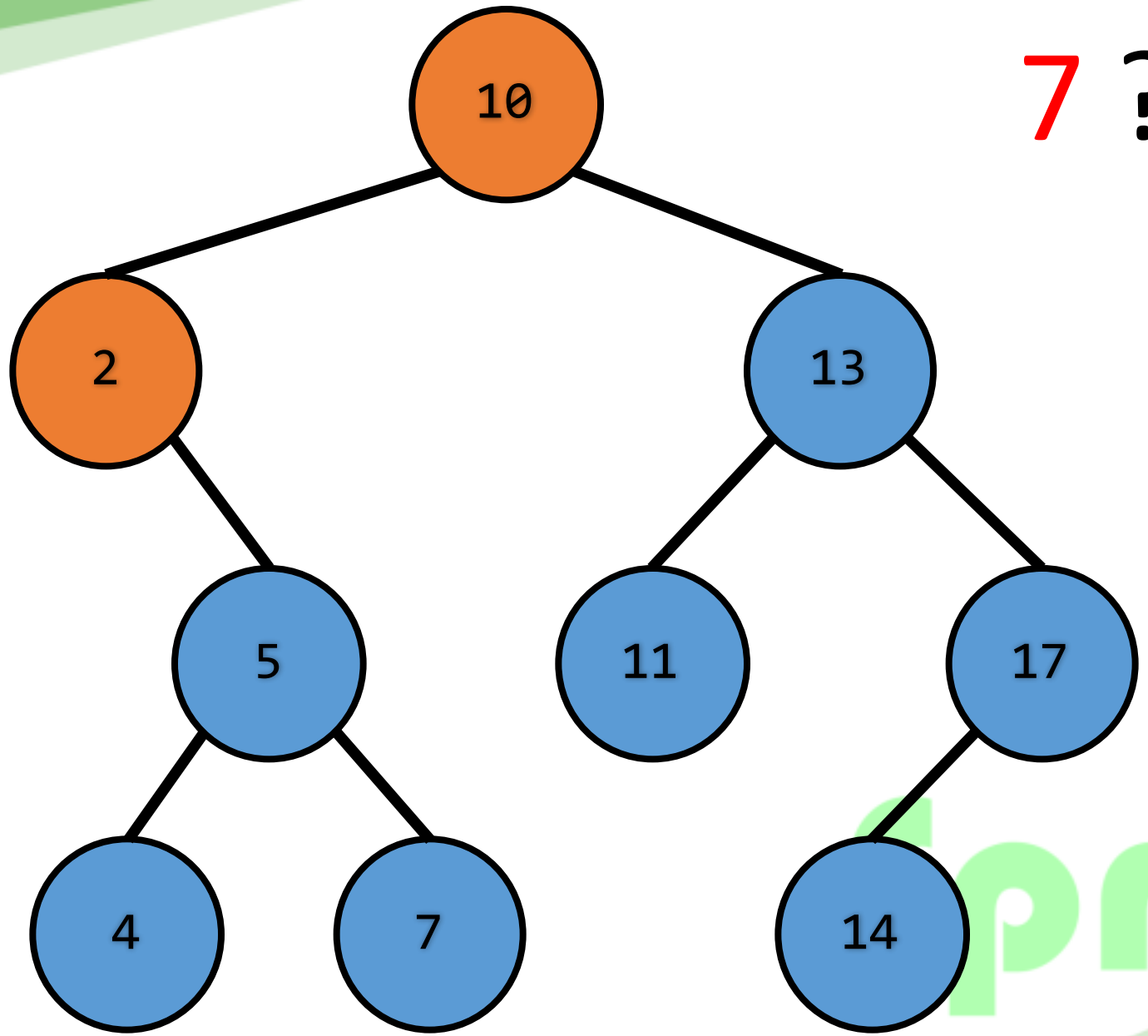

10
2
13
5
11
17
4
7
14



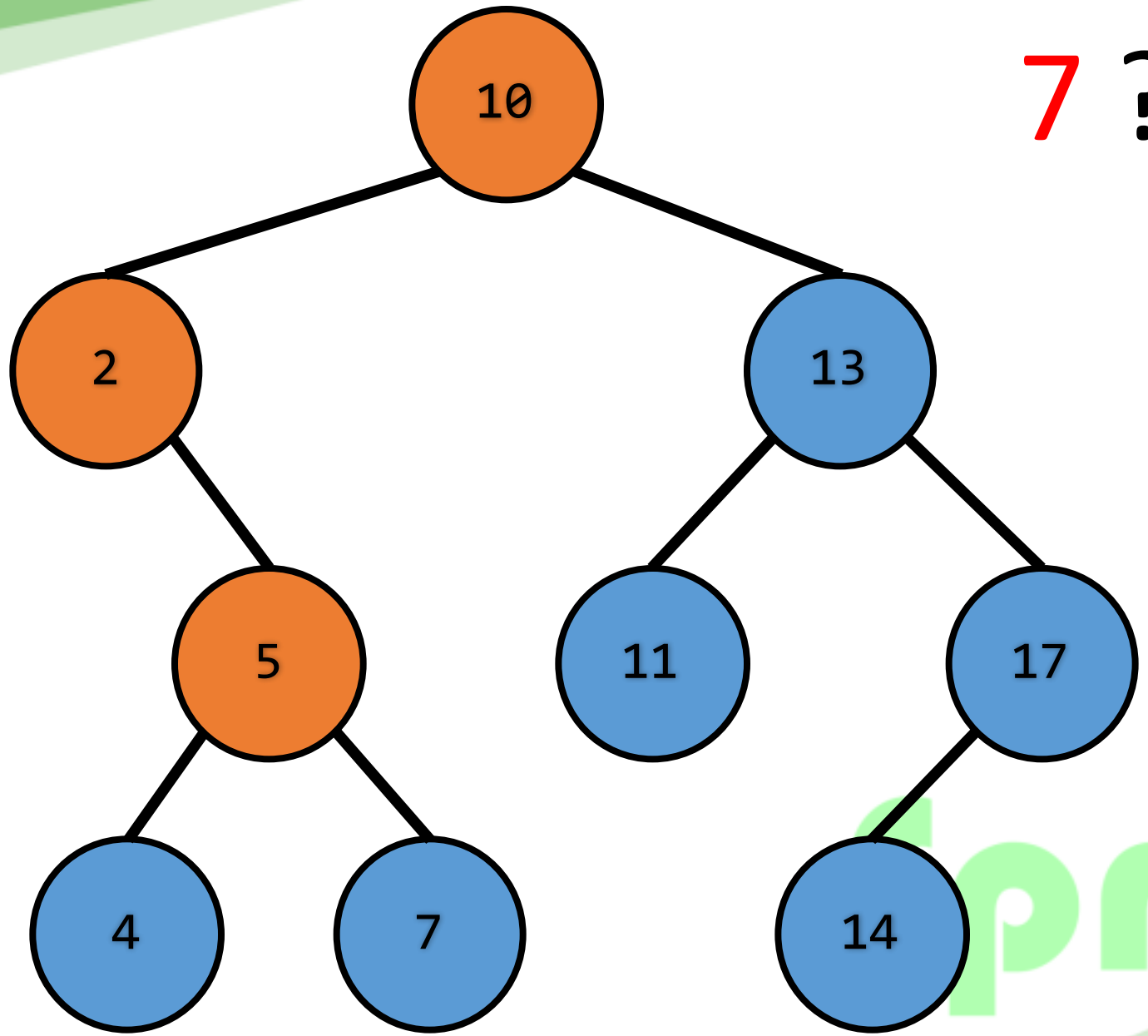
7 ?



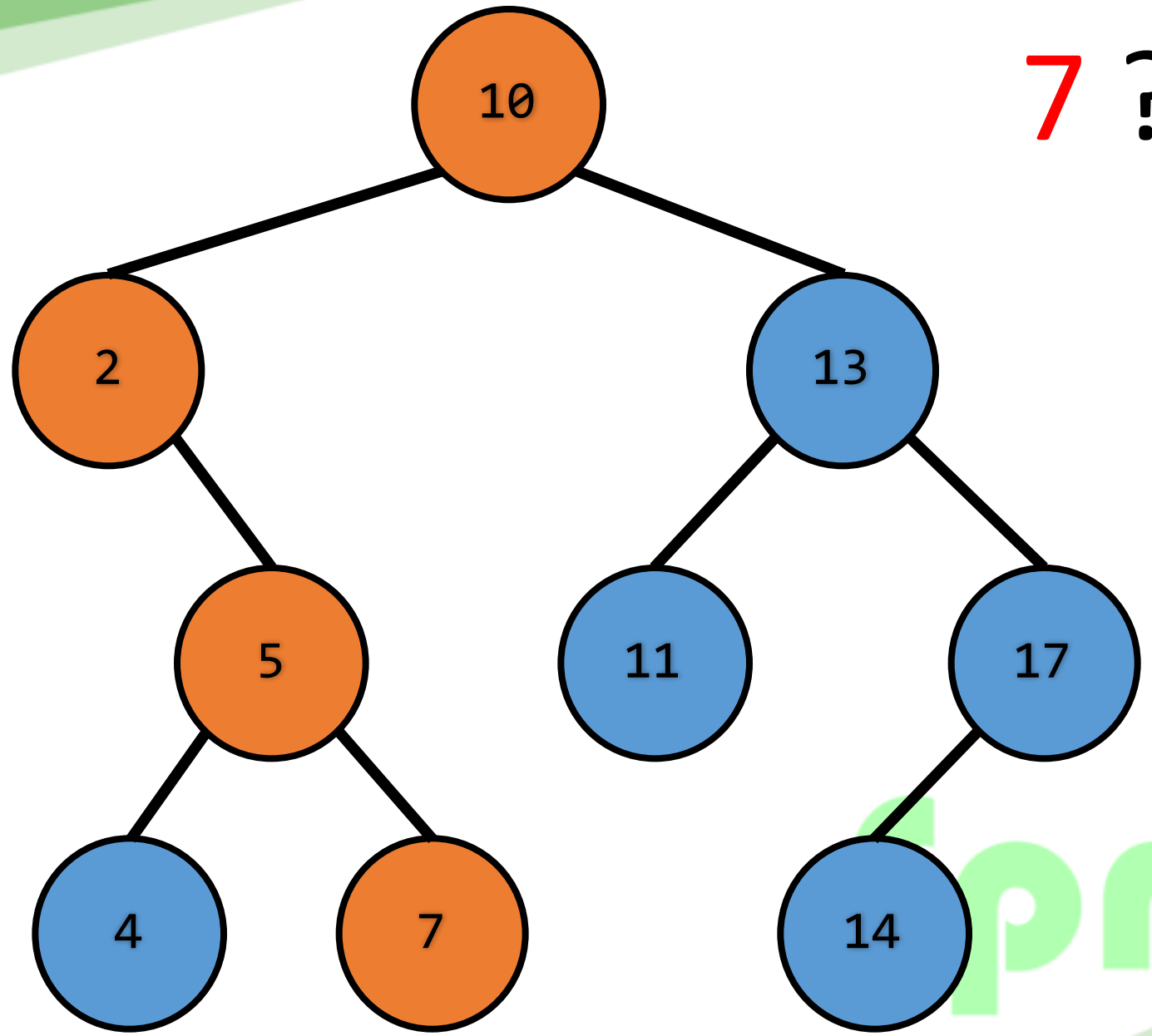
7 ?



7 ?



7 ?

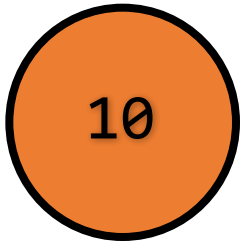


7 ?

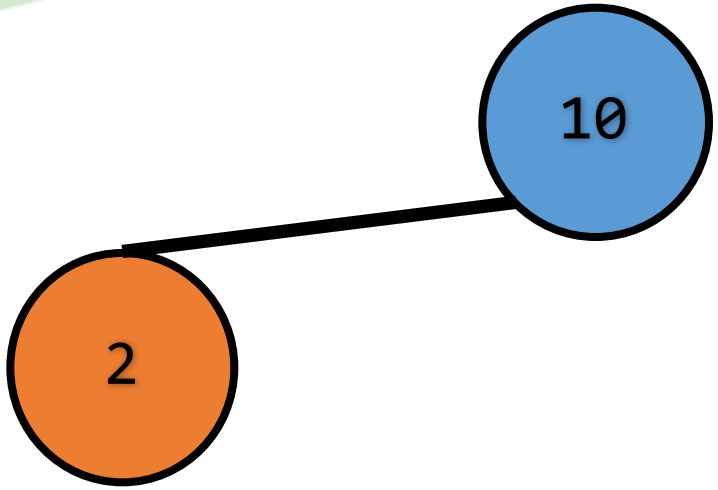
10,2,5,13,17,11,7,14,4

Sprout

10,2,5,13,17,11,7,14,4

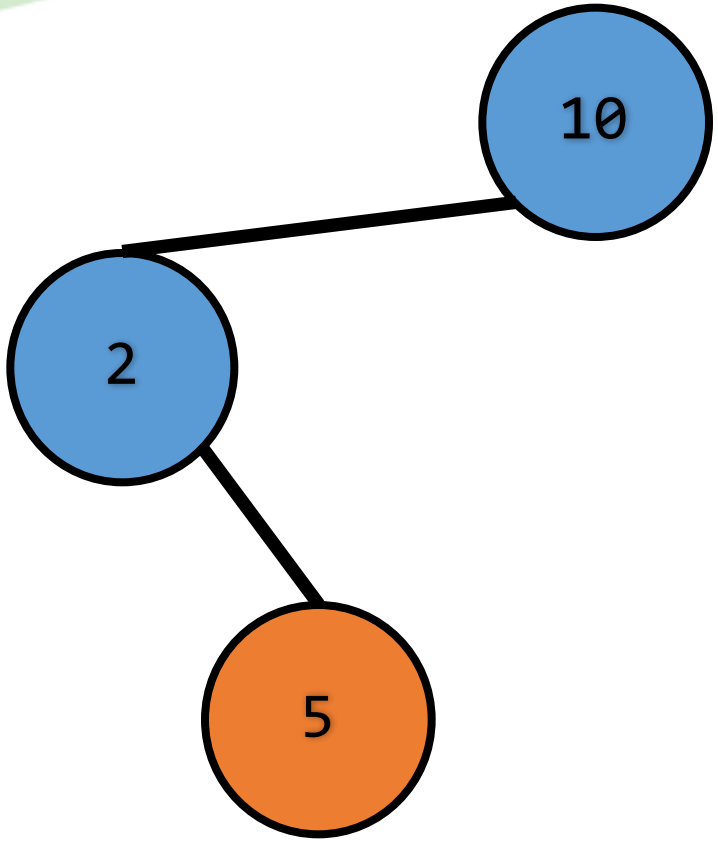


10,2,5,13,17,11,7,14,4



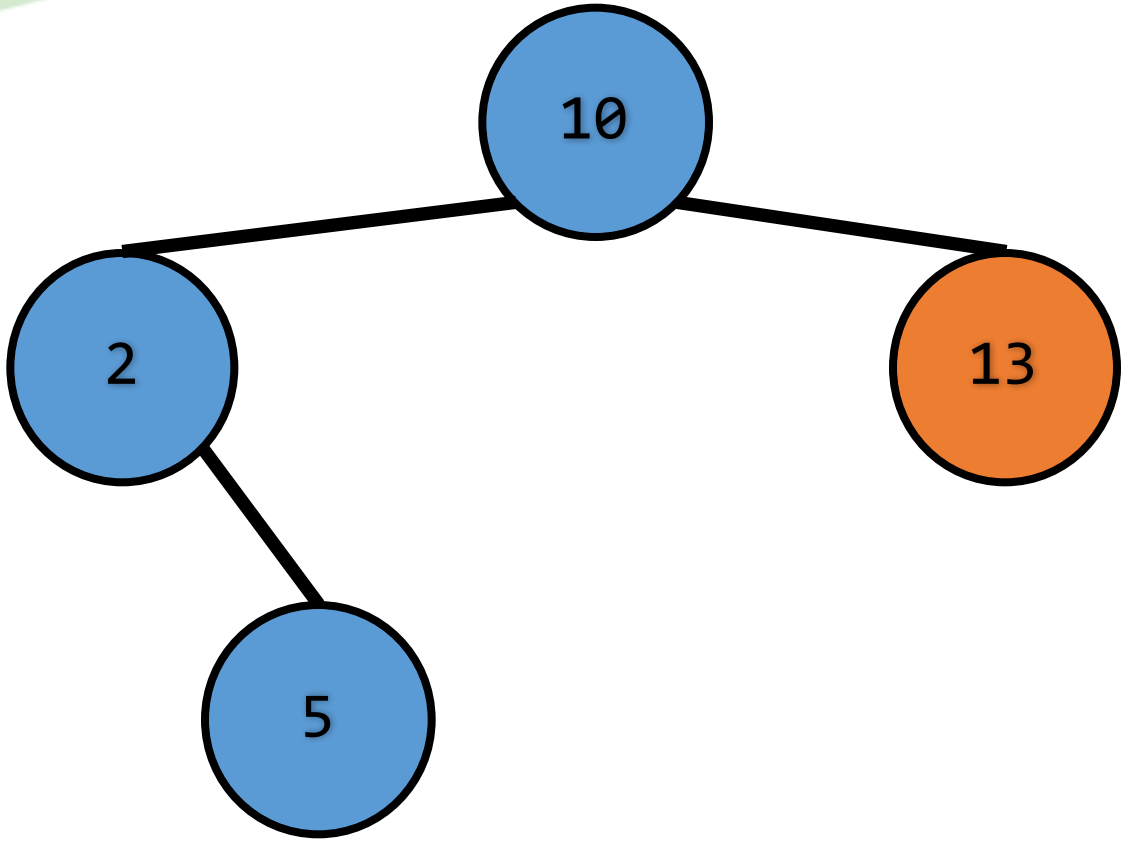
Sprout

10,2,5,13,17,11,7,14,4



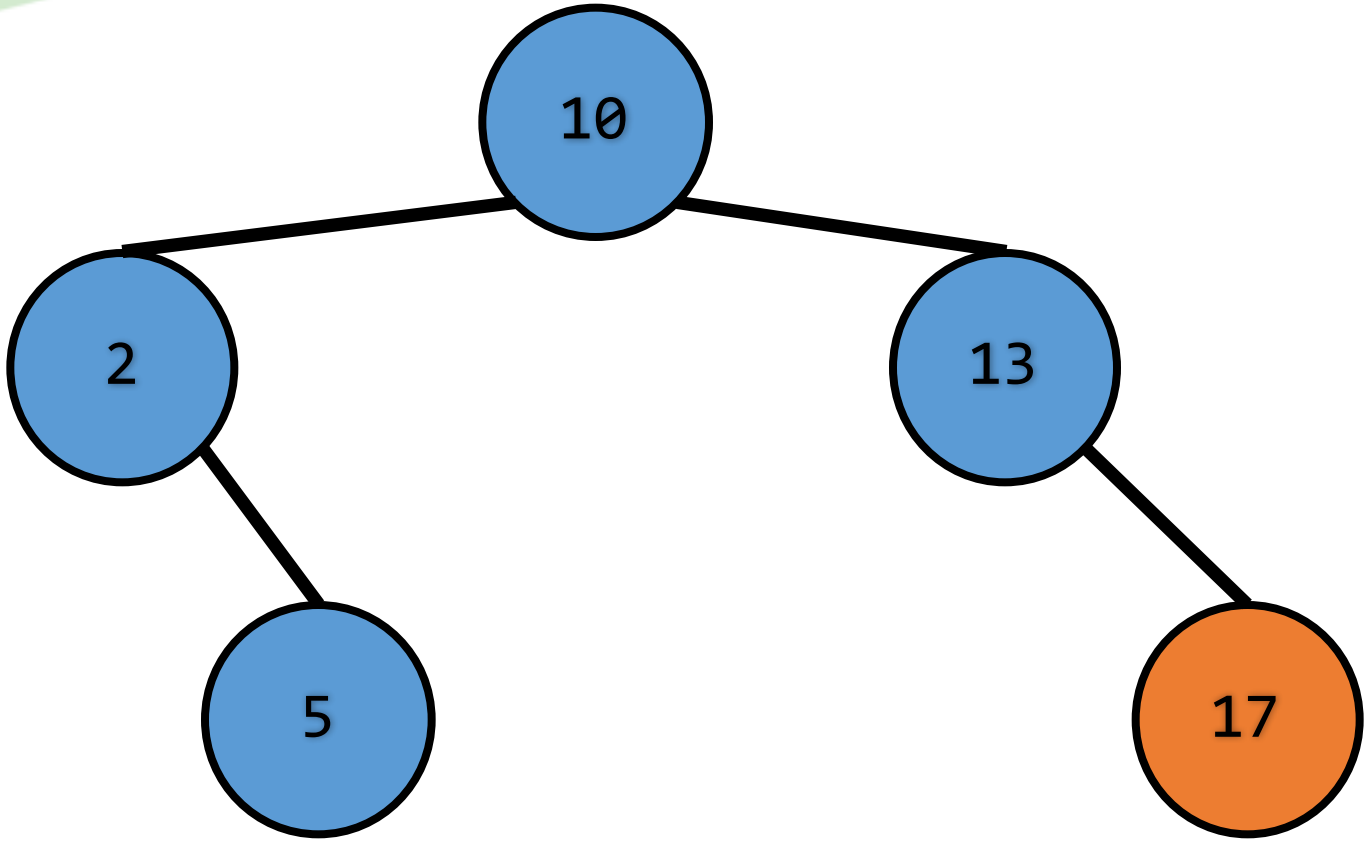
Sprout

10,2,5,13,17,11,7,14,4

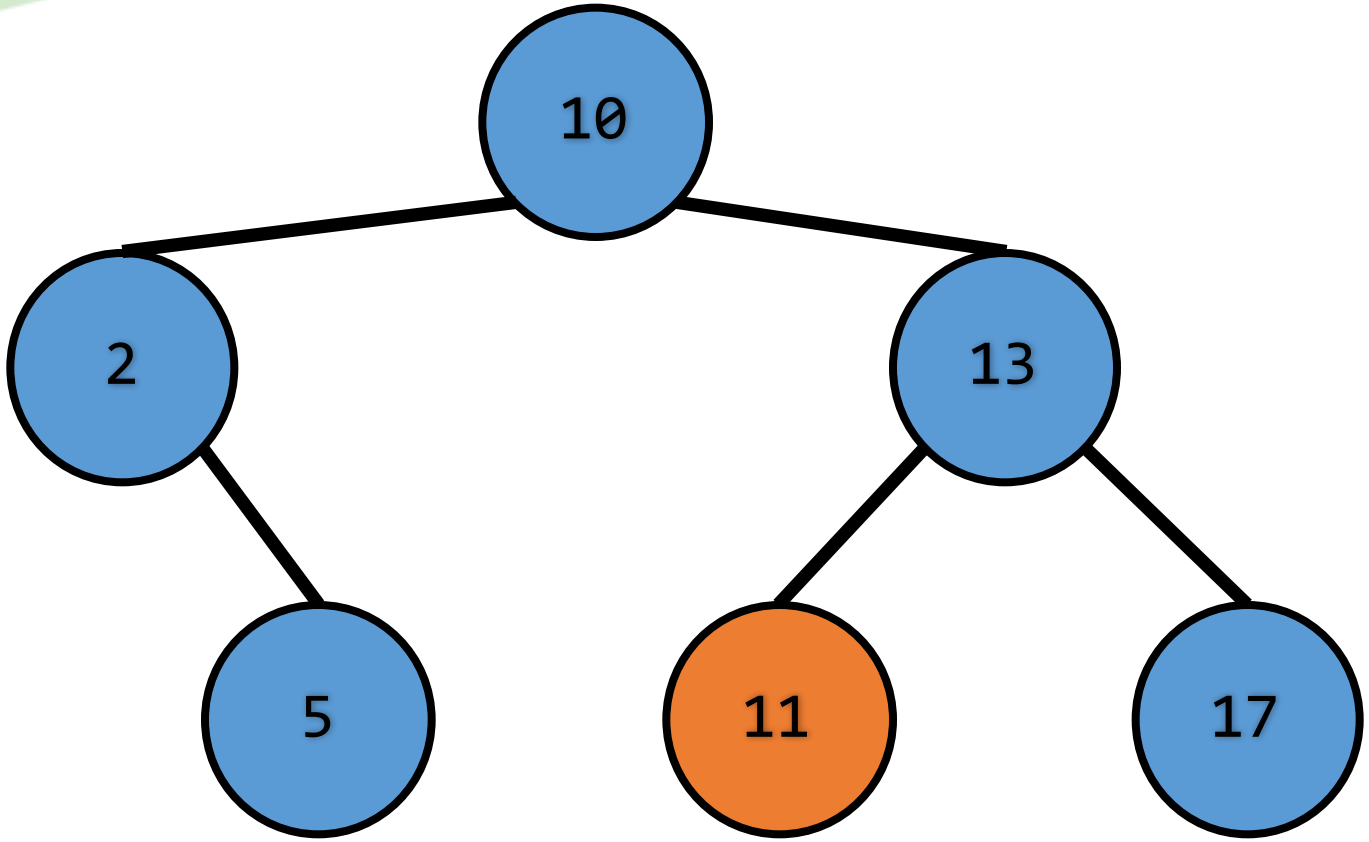


Sprout

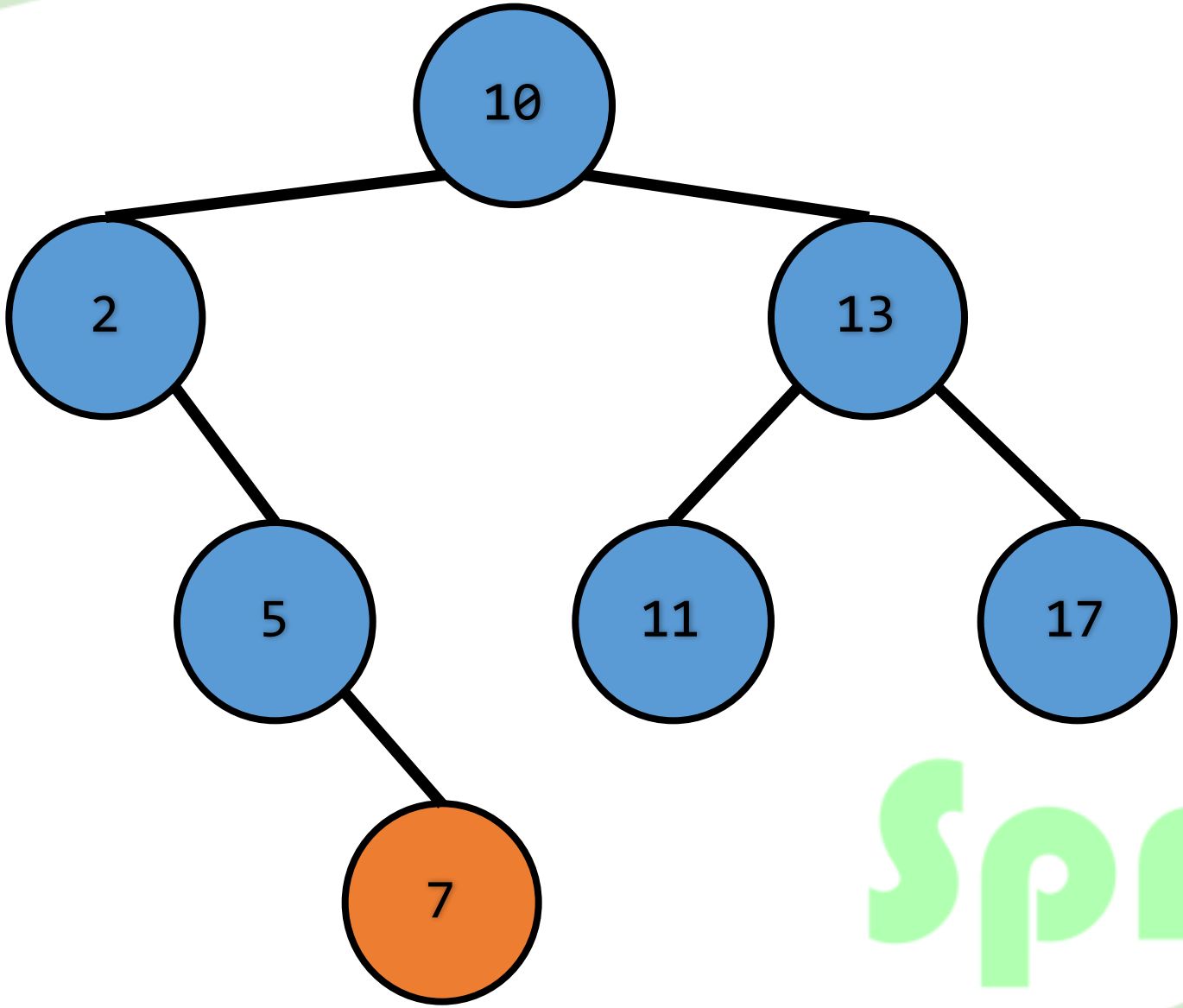
10,2,5,13,17,11,7,14,4



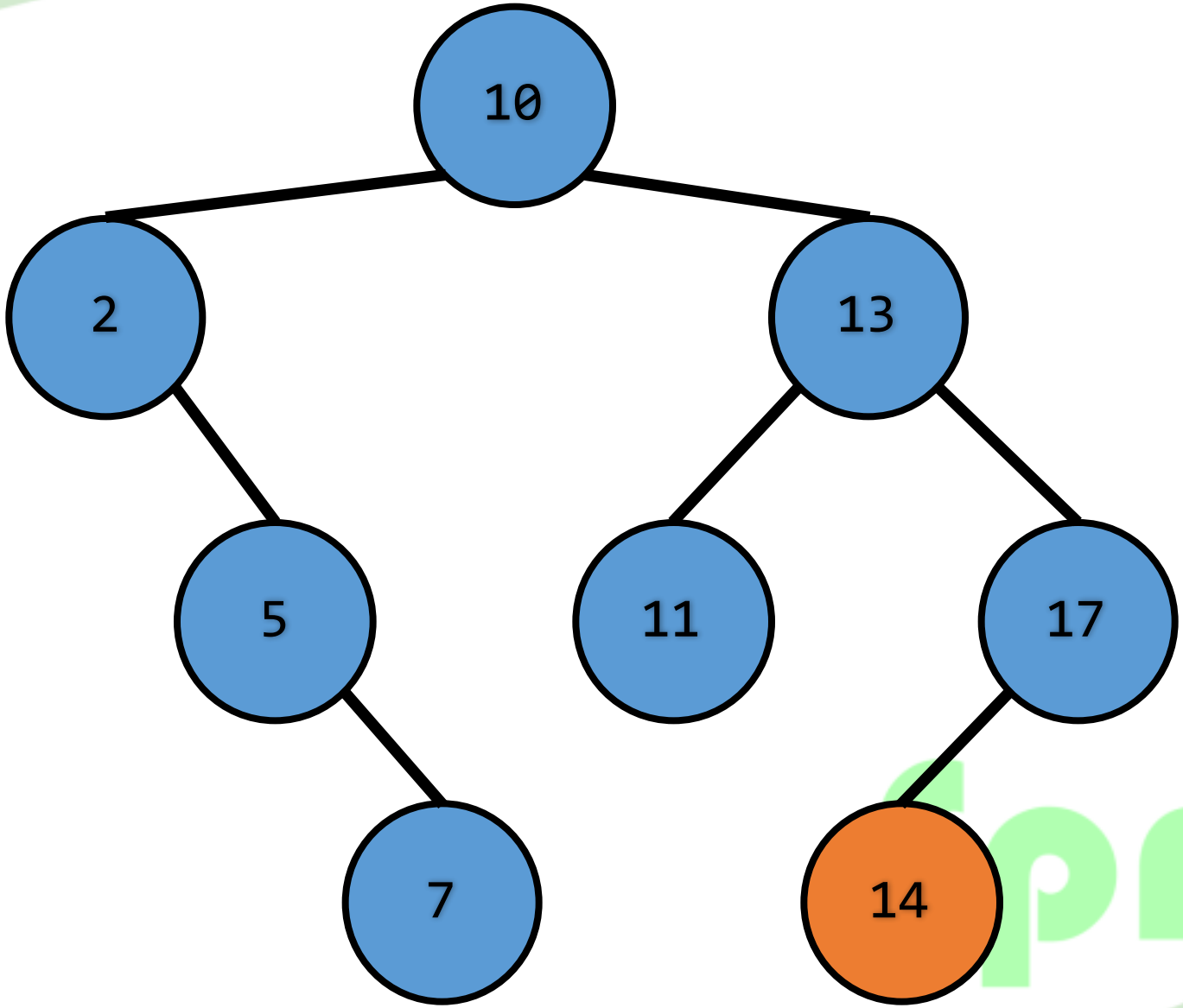
10,2,5,13,17,11,7,14,4



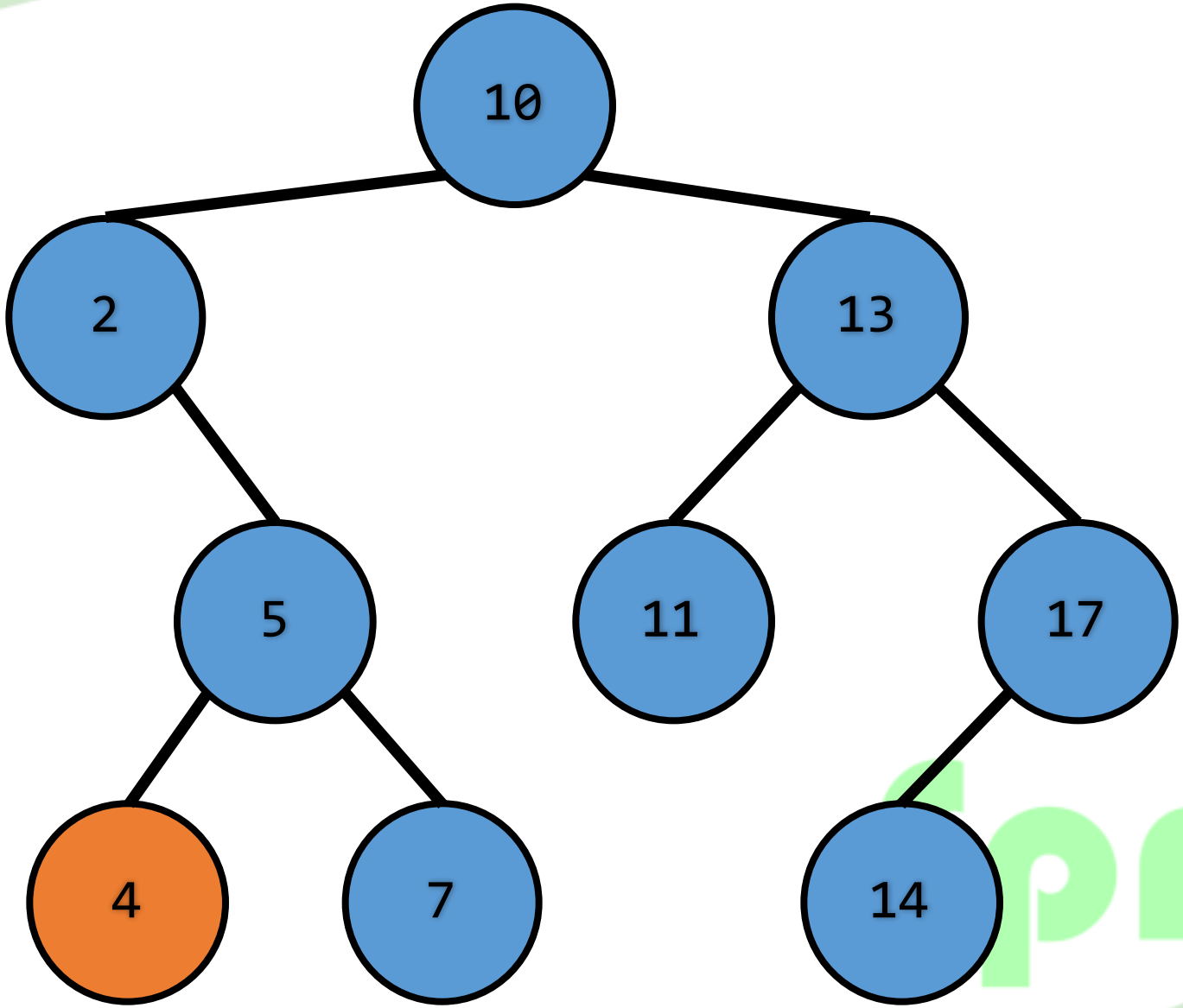
10,2,5,13,17,11,7,14,4



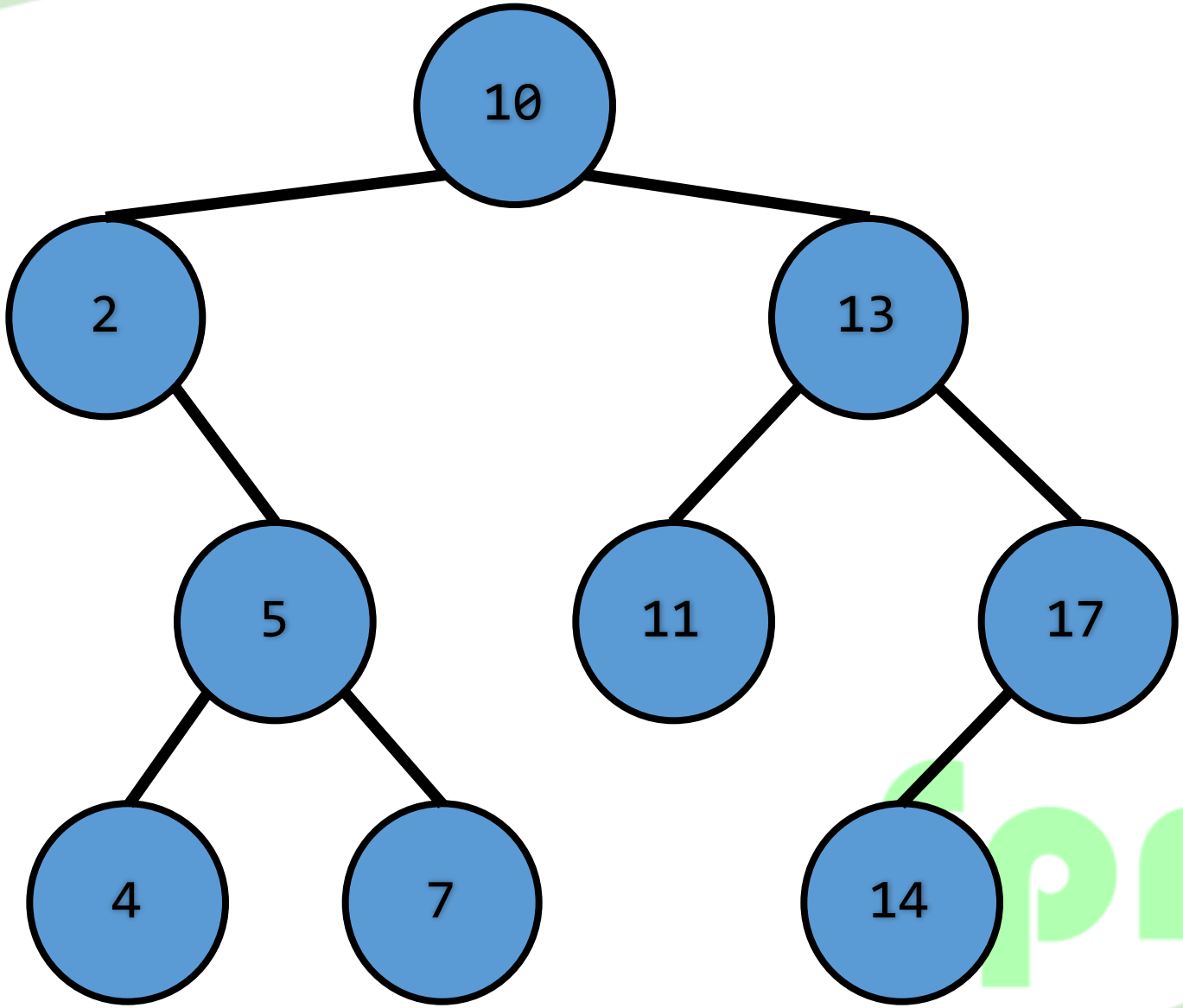
10,2,5,13,17,11,7,14,4



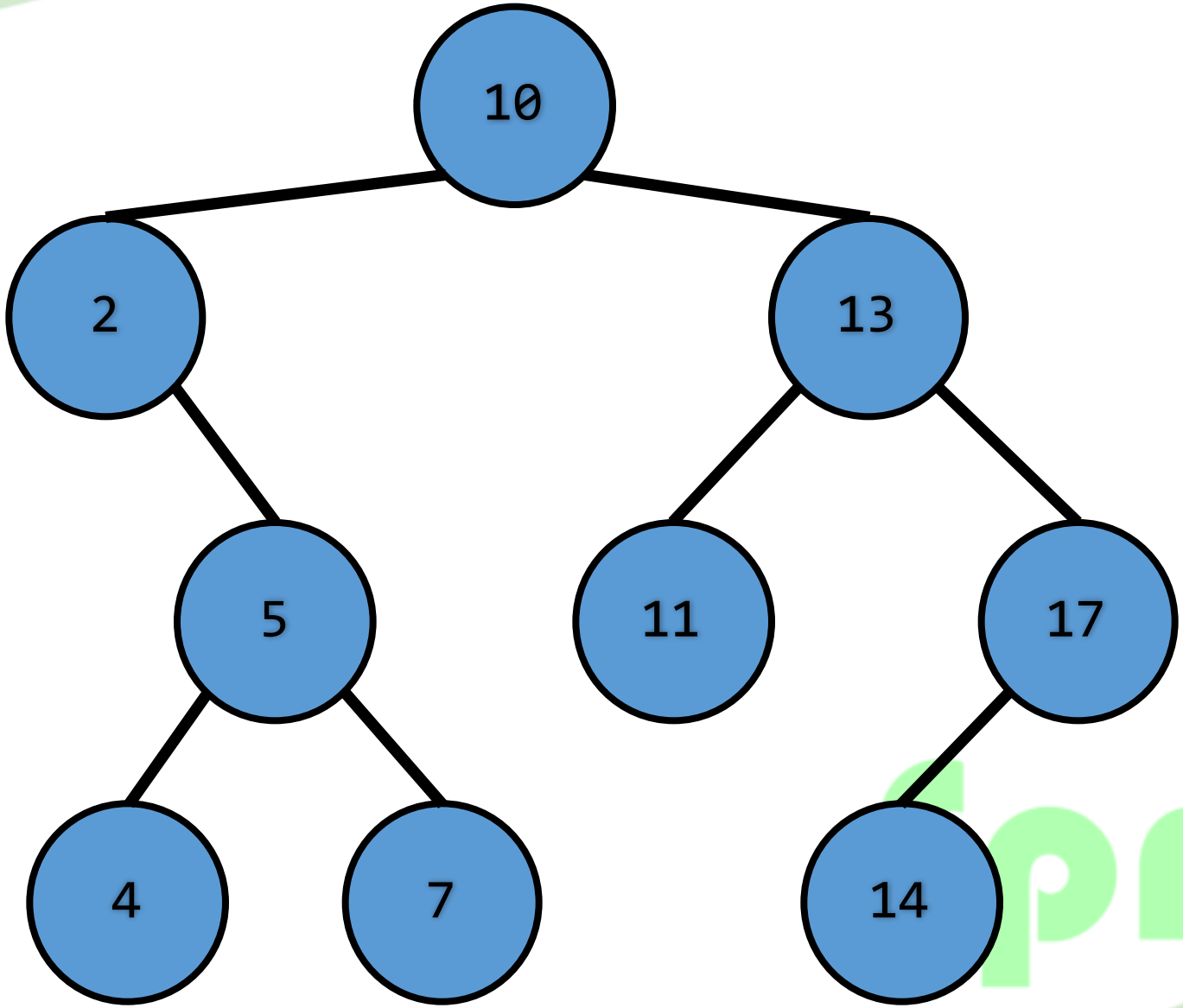
10,2,5,13,17,11,7,14,4



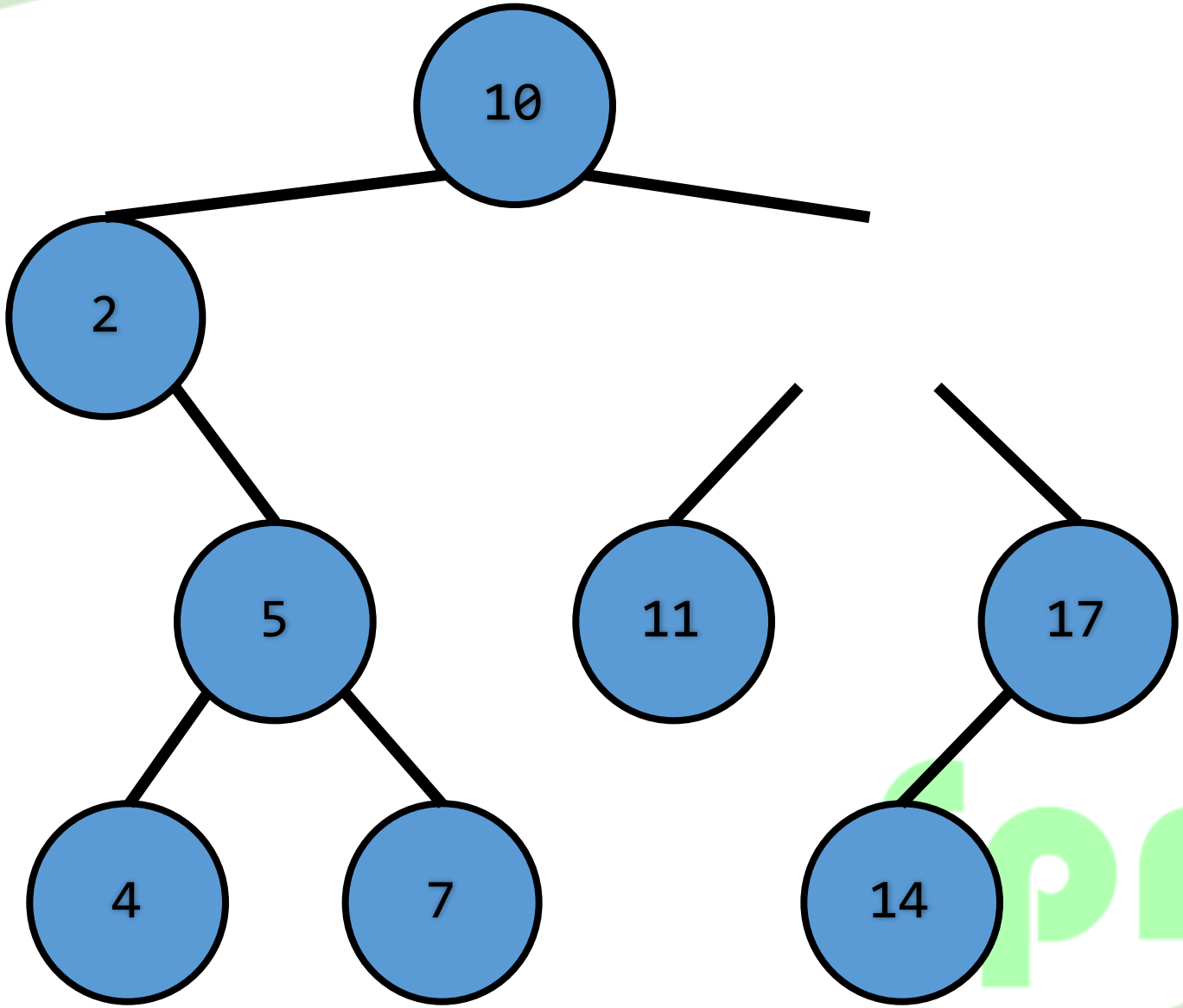
10,2,5,13,17,11,7,14,4



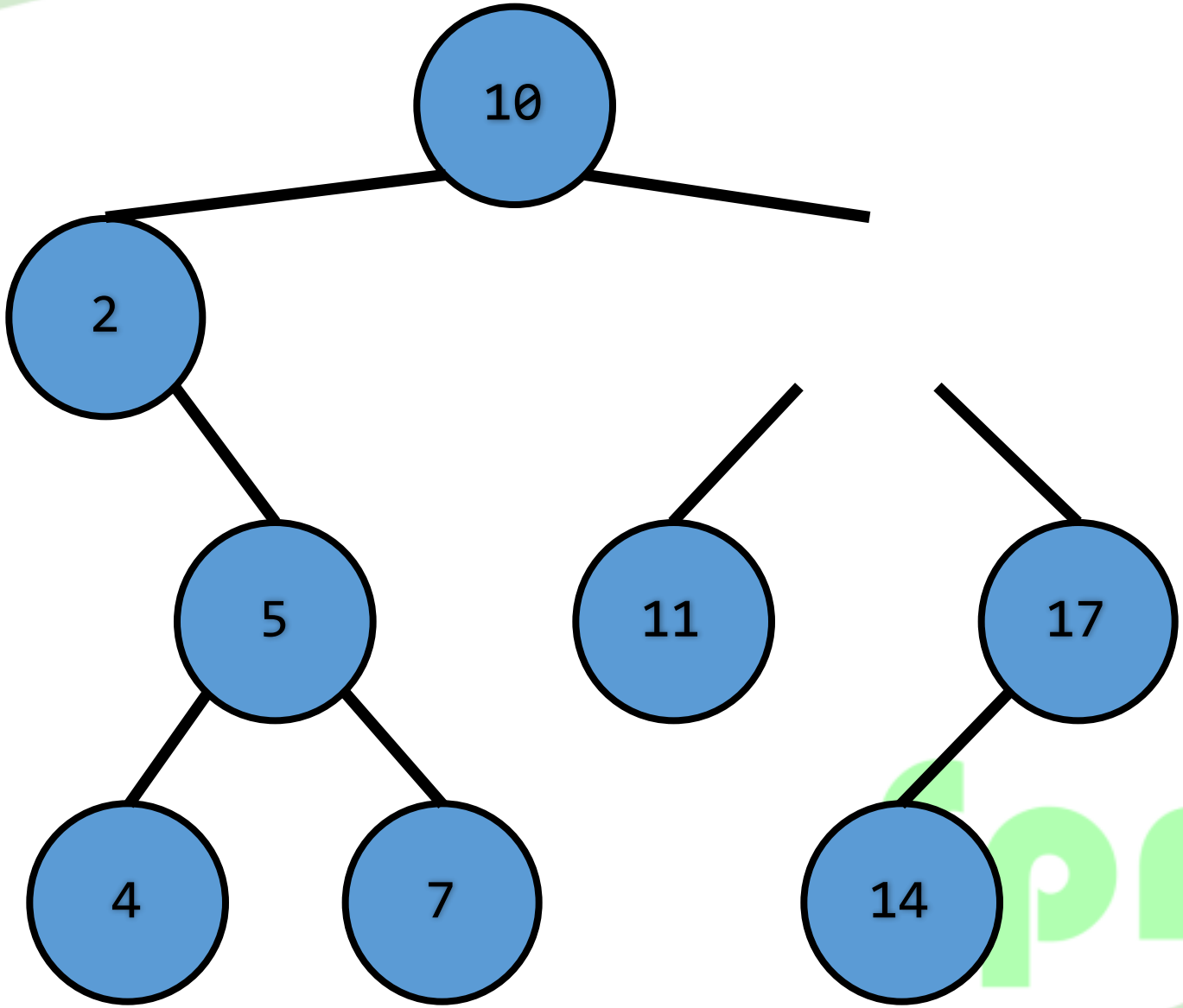
Delete 13



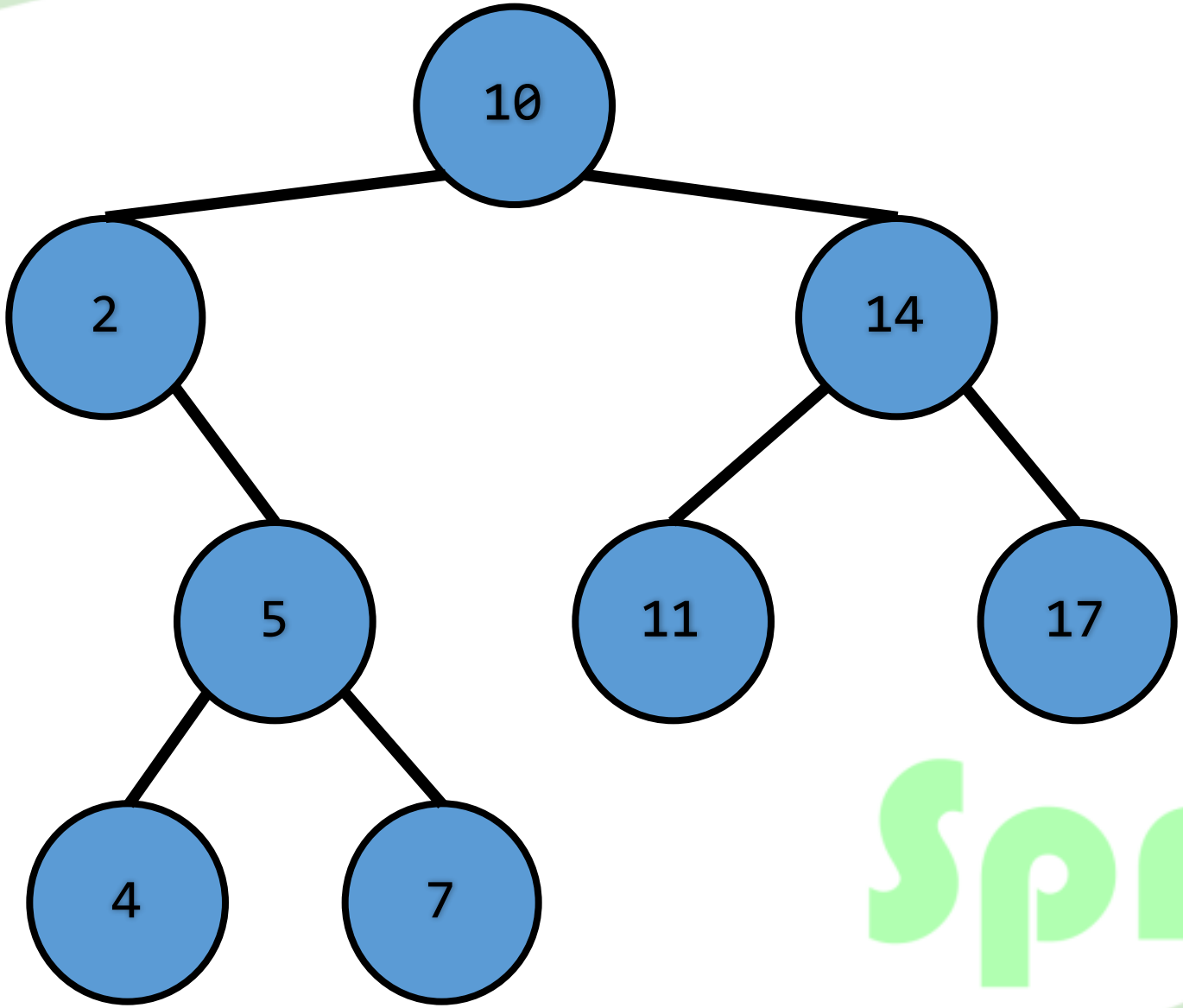
Delete 13



Delete 13



Delete 13



時間複雜度?

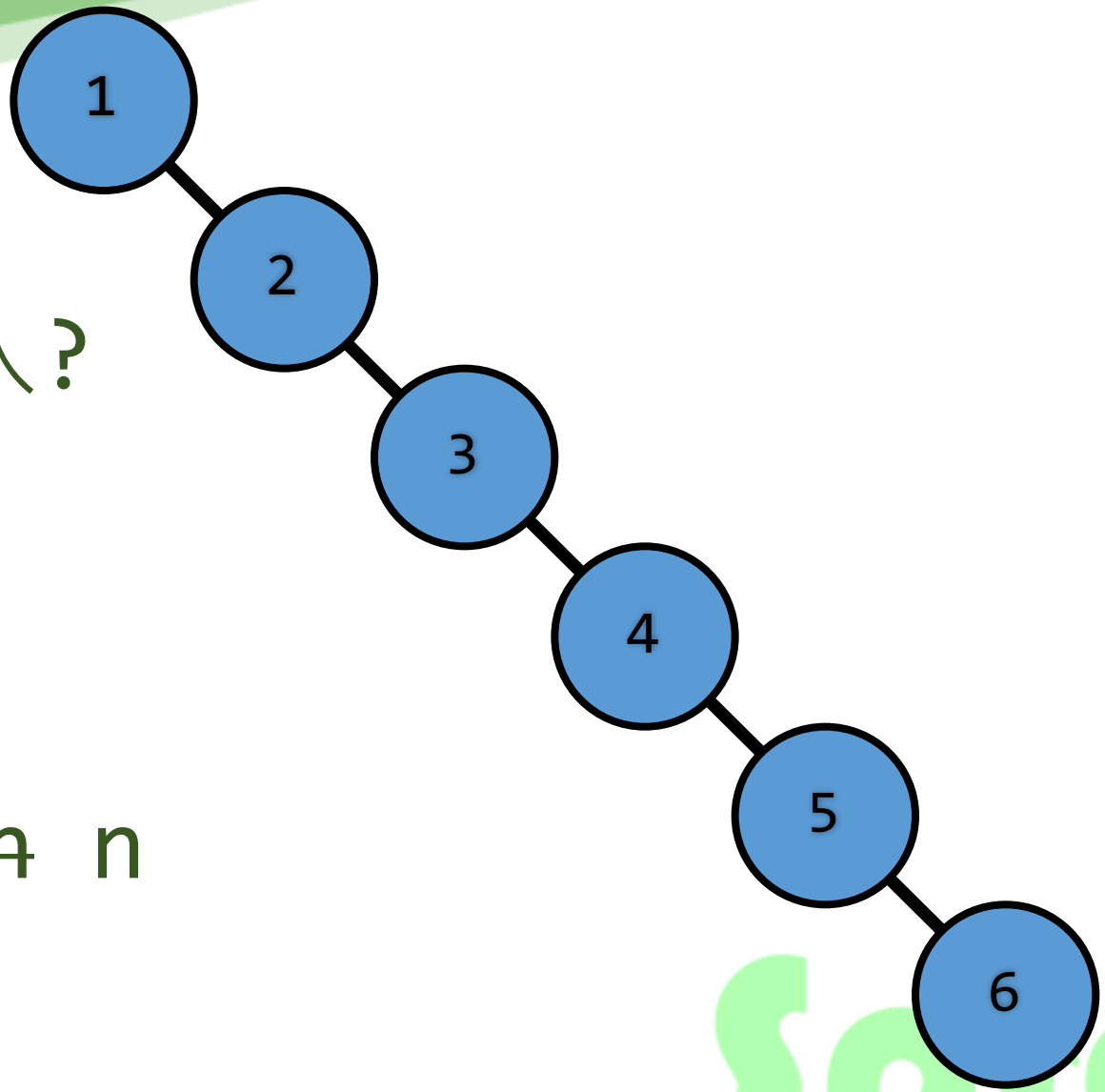
- 新增節點
  - O(h) ，h是樹的深度
- 搜尋一個值
  - O(h) ，h是樹的深度
- 刪除一個值
  - O(h) ，h是樹的深度
- 二元樹有n個節點的時候，深度就是O(log n)
  - ?

Sprout

時間複雜度?

- 新增節點
  - O(h) ，h是樹的深度
- 搜尋一個值
  - O(h) ，h是樹的深度
- 刪除一個值
  - O(h) ，h是樹的深度
- ~~二元樹有n個節點的時候，深度就是O(log n)~~
  - ?

Sprout



用什麼順序插入?

1,2,3,4,5,6

深度 -> ~~log n~~ n