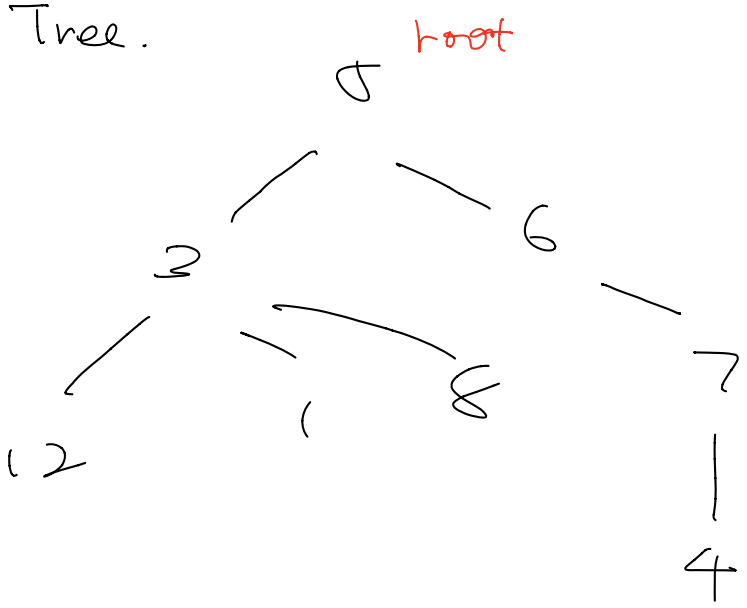
Tree.



Tree: value
children: [Tree]

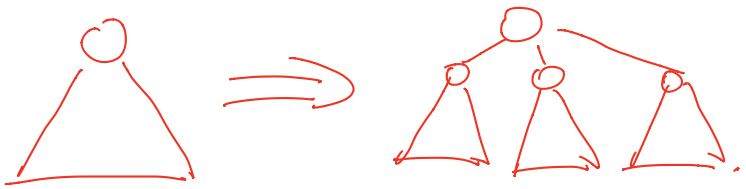
root.
leaf: no children
internal node: ≥1 children

height: (node heights) 4
longest path length: 3

depth: level

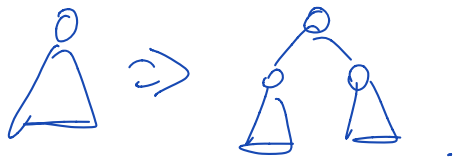
arity: 3

(6)



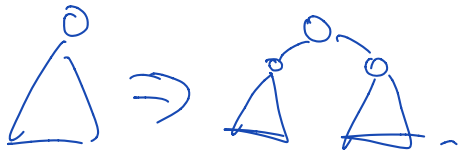
```
def count(t):
    acc = 1
    for c in t.children:
        acc += count(c)

    return acc
```

```
def count_leaves(t):
    if t.children == []:
        return 1

    else:
        acc = 0
        for c in t.children:
            acc += count_leaves(c)
        return acc
```

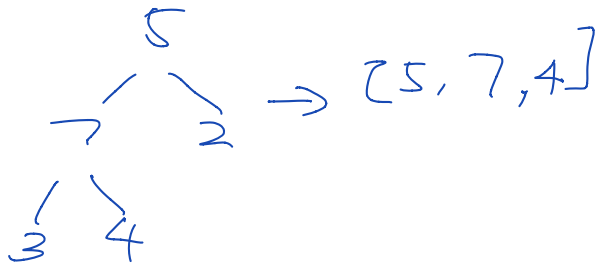


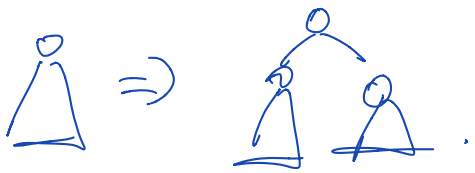
```
def height(t):
    if len(t.children) == 0:
        return 1

    else:
        acc = []
        for c in t.children:
            acc.append(height(c))
        return max(acc) + 1

max([height(c) for c in t.children]) + 1
```

```
def gather_odd(t):
    acc = []
    if t.value % 2 == 1:
        acc.append(t.value)
    for c in t.children:
        acc.extend(gather_odd(c))
    return acc
```



```
def get_longest_path(t):
    if not t.children:
        return [t.value]
    else:
        acc = []
        for c in t.children:
            path = get_longest_path(c)
            if len(path) >= len(acc):
                acc = path
        return [t.value] + acc
```





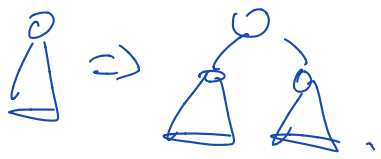
```
def get_all_path(t):
    if not t.children:
        return [[t.value]]
    else:
        acc = []
        for c in t.children:
            paths = get_all_path(c)
            for path in paths:
                acc.append([t.value] + path)
        return acc
```

t → [[5,2],[5,7]]

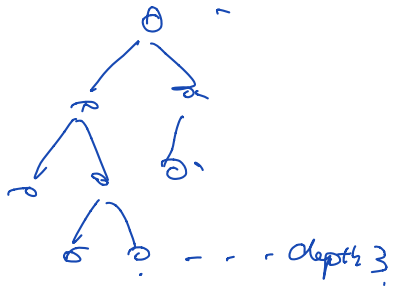
```
def get_path_to(t, value):
    if t.value == value:
        return [t.value]
    else:
        if not t.children:
            return []
        else:
            for c in t.children:
                result = get_path_to(c, value)
                if len(result) > 0:
                    return [t.value] + result
        return []
```

[5,3,7]

[]



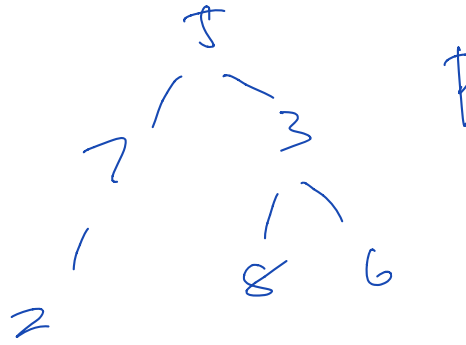
```
def count_at_depth(t, d):
    if d == 0:
        return 1
    if not t.children:
        return 0
    else:
        acc = 0
        for c in t.children:
            acc += count_at_depth(c, d - 1)
        return acc
```



Traversal.

pre order root 在前

past order. root 在后.



pre 5 7 2 3 8 6

post. 2 7 8 6 3 5

Level 5 7 3 2 8 6

queue. 2 8 6

5 7 3 2 8 6

1. subtree 有沒有?

2. contains.

[ ].

X X X