一般是暴力搜索 每个节点都遍历一遍

搜索

如果遍历的数据结构本身没有任何特点，就遍历所有的节点，同时保证
- 每个点访问一次，
- 且仅访问一次
最后找到结果

在 树\图 中寻找特定结点

对于节点的访问顺序不限，可分为
- 深度优先 depth first search
- 广度优先 breadth first search

深度优先 DFS

广度优先 —— 适合算深度、层级问题

DFS 代码 - 非递归写法

```
def DFS(self, tree):

    if tree.root is None:
        return []

    visited, stack = [], [tree.root]

    while stack:
        node = stack.pop()
        visited.add(node)

        process (node)
        nodes = generate_related_nodes(node)
        stack.push(nodes)

    # other processing work
```

示例代码

```
def dfs(node):

    if node in visited:
      # already visited
      return

    visited.add(node)

    # process current node
    # ... # logic here
    dfs(node.left)
    dfs(node.right)
```

DFS 代码 - 递归写法

```
visited = set()

def dfs(node, visited):
    visited.add(node)
    # process current node here.
    ...
    for next_node in node.children():
        if not next_node in visited:
            dfs(next node, visited)
```

BFS 代码

```
def BFS(graph, start, end):

    queue = []
    queue.append([start])
    visited.add(start)

    while queue:
        node = queue.pop()
        visited.add(node)

        process(node)
        nodes = generate_related_nodes(node)
        queue.push(nodes)

    # other processing work
    ...
```