P4 练习参考速查表（Cheat Sheet）

---

1. 广度优先搜索（BFS）

标准实现

```python
from collections import deque

def bfs(graph, start):
    """
    Breadth-First Search

    Parameters:
        graph: Graph represented as adjacency list
        start: Source vertex

    Returns:
        distances: Dictionary, distances[v] is distance from start to v
        parents: Dictionary, parents[v] is parent vertex in shortest path
    """
    # Initialize
    distances = {}
    parents = {}
    visited = set()

    # Initialize source vertex
    distances[start] = 0
    parents[start] = None
    visited.add(start)

    # Use queue to store current level vertices
    queue = deque([start])

    # BFS main loop
    while queue:
        u = queue.popleft()  # FIFO: first in, first out

        # Traverse all neighbors of u
        for v in graph.get(u, []):
            if v not in visited:
                # Discover new vertex
                visited.add(v)
                distances[v] = distances[u] + 1
                parents[v] = u
                queue.append(v)

    # Set distance to infinity for unvisited vertices
    for v in graph:
        if v not in distances:
            distances[v] = float('inf')
```

```
45
46      return distances, parents
```

路径重构

```
1  def reconstruct_path(parents, start, target):
2      """
3      从parent字典重构从start到target的路径
4
5      Parameters:
6          parents: BFS返回的parent字典
7          start: 起始顶点
8          target: 目标顶点
9
10     Returns:
11         路径列表，如果不存在路径返回None
12     """
13     path = []
14     current = target
15     while current is not None:
16         path.append(current)
17         current = parents.get(current)
18     path.reverse()
19     return path if path[0] == start else None
```

关键点

- 使用 deque 实现队列（FIFO）
- queue.popleft(): 从左侧出队
- queue.append(): 从右侧入队
- 时间复杂度: $O(|V| + |E|)$
- 空间复杂度: $O(|V|)$

2. 深度优先搜索（DFS）

标准实现（使用 visited set）

```python
def dfs(graph, start):
    """
    Depth-First Search

    Parameters:
        graph: Graph represented as adjacency list
        start: Source vertex

    Returns:
        parents: Dictionary, parents[v] is parent vertex in DFS tree
    """
    parents = {start: None}
    visited = set()

    def visit(u):
        visited.add(u)
        for v in graph.get(u, []):
            if v not in visited:
                parents[v] = u
                visit(v)  # Recursive call

    visit(start)
    return parents
```

简化版本（不使用 visited set）

```python
def dfs_simple(graph, start):
    """
    DFS简化版本：使用parents.keys()代替visited set
    """
    parents = {start: None}

    def visit(u):
        for v in graph.get(u, []):
            if v not in parents:  # 检查是否已访问
                parents[v] = u
                visit(v)

    visit(start)
    return parents
```

关键点

- 递归实现，沿着路径深入探索
- 可以使用 `parents.keys()` 代替 `visited` 集合
- 时间复杂度：$O(|E|)$
- 空间复杂度：$O(|V|)$（递归栈）