记忆化搜索

使用记忆化搜索的时间复杂度相对树形 dp 要高很多，但是代码相对简单。

O(n^2)

Python

```python
# 通用记忆化搜索模板，使用Python标准库
# 邻接表表示图
graph = {
    1: [2, 3],
    2: [1, 4, 5],
    3: [1],
    4: [2],
    5: [2]
}

# 记忆化数组，存储每个节点的搜索结果
memo = {}

# 记忆化搜索的DFS函数
def dfs(node, parent):
    # 如果当前节点的结果已经计算过，直接返回缓存的结果
    if node in memo:
        return memo[node]

    # 初始化结果
    result = 0

    # 遍历当前节点的所有相邻节点
    for neighbor in graph[node]:
        if neighbor != parent:  # 防止返回到父节点
            result += dfs(neighbor, node)

    # 将当前节点的结果存入memo缓存
    memo[node] = result + 1  # +1 表示处理当前节点

    return memo[node]

# 记忆化搜索只能计算出当前节点node的最优值；
# 所以如果要计算整个图，需要对每个节点使用记搜，并比较结果

min_path_sum = INF
for node in range(1, n + 1):
    # 清空memo，因为每次起点不同
    memo.clear()
    current_sum = dfs(node, -1)
    if current_sum < min_path_sum:
        min_path_sum = current_sum
        start = node
```