第十一章 递推算法

1. 基本思想

递推算法是一种通过将问题分解为较小的子问题来解决问题的方法。 递推算法的基本思路是，从 初始状态开始，通过一系列推导和选代，逐步得到问题的解。在每一步推导中，利用已知的解或经得到的中间结果，来计出下一步的解。

2. 递推步骤

1. 确定初始状态：递推算法是从已知条件推导出未知结果的过程， 因此需要确定问题的初始条件 这个初始条件可以是问题中给出的初始值，也可以是手动求解的一些值
2. 明确问题的递推关系：递推算法的核心就是通过已知条件推导出未知结果， 因此需要通过观察问题的特点， 我到问题的递推关系，也就是递推式
3. 迭代计算：根据问题的递推关系和初始条件， 可以开始逐步计算得到未知结果。 递推计算的过程就是反复应用递推关系，将已知的结果代入递推关系中得到新的结果，然后再将新的结果代入递推关系中得到更新的结果，以此类推， 直到问题解决
4. 终止条件：确定算法的终止条件，即满足终正条件时算法结束。

3. 递推方法的分类

1. 顺推法：从已知条件出发，逐步推算出要解决的问题的方法
2. 逆推法：从已知问题的结果出发，逐步推算出问题的开始条件。

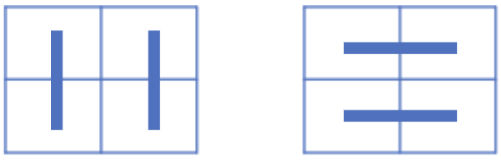
4.顺推法示例

有一块 2*n的空地，现要使用n块1*2 的地砖铺满这块空地，请问有多少种不同的铺法?

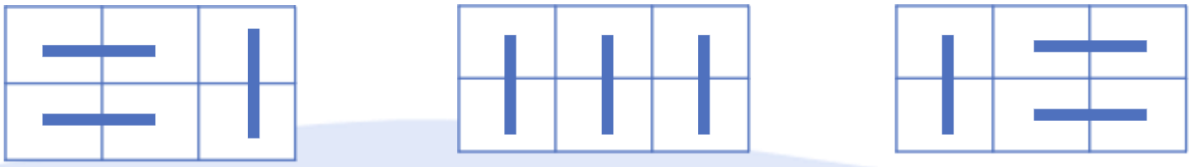
分析：

假设 n = 1，铺满 2 * 1 的空地只有 1 种铺法；

假设 n = 2，铺满 2 * 2 的空地有 2 种不同铺法；



假设 n = 3，铺满 2 * 3 的空地有 3 种不同铺法；



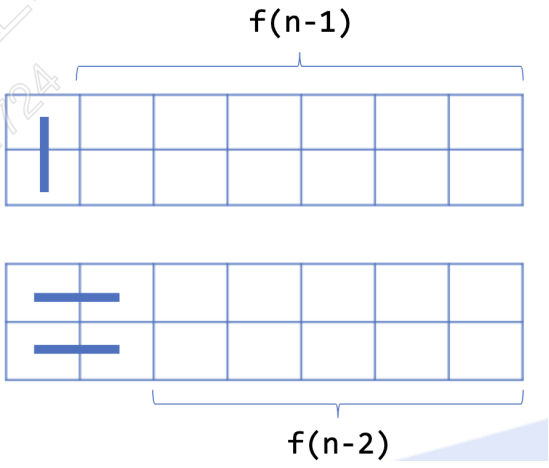
假设 2 * n 的空地有 f(n) 种铺法；

若第一个块地砖竖排放置，剩下有 n - 1 列空地需要排列，这时铺法数为 f(n - 1)；

若第一个块地砖横排放置，剩下有 n - 2 列空地需要排列，这时铺法数为 f(n - 2)；

所以有：

$$\begin{cases} f(1) = 1 \\ f(2) = 2 \\ f(3) = 3 \\ f(n) = f(n-1) + f(n-2), \quad n > 3 \end{cases}$$



1. 初始条件为 f(1) = 1, f(2) = 2;
2. 递推关系式为 f(n) = f(n-1) + f(n-2)

代码示例：

```cpp
#include<iostream>
using namespace std;
int f[100];
int func(int n)
{
    if(n<=2) return n;//当n小于等于2都返回
    f[1] = 1;
    f[2] = 2;
   //从3开始 递归过程
    for(int i = 3 ;i <= n;++i)
        f[i] = f[i-1]+f[i-2];
    return f[n];
}
int main()
{
    int n;
    cin >> n;
    int ways = func(n);
    cout << ways;
    return 0;
}
```

5.逆推法示例

树上有若干个桃子，第一天猴子吃掉树上桃子的一半多1个，第二天吃掉剩余桃子的一半多 个，以此类推，每一天都吃掉剩余桃子的一半多1个，第七天吃完后，树上还剩下1个桃子 问树上原来有多少个桃子?

- 分析 这是一个逆序递推的问题，我们可以从第七天逆推到第一天， 根据题目描述的递推关系式计算出树上原来的桃子数量
  根据题目描述， 每天都会吃掉剩余桃子的一半多一个，设peach（n）表示第n天吃完后树 上剩余的桃子数量，peaches （n+1） 表示第 n+1 天吃完后树上的桃子数量 有 peaches （n+1）=peaches（n）/2 -1
  可以得出递推关系式： peaches （n）=（peaches （n+1）+1）*2，
  初始状态： peaches （7）=1

- 分析
  可以使用一个变量来保存当前的桃子数量，并在每次送代中进行更新。 首先， 我们知道第七天吃 完后树上剩下1个桃子，所以我们可以将初始桃子数量设为 1。然后，我们从第七天开始逆序选代到第0天（第一天未吃之前的状态）。在每次选代中，根据递推关系式计算出前一天吃完 后剩余的桃子数量，并将其保存到变量中。最后，我们输出第0天的桃子数量即可

代码示例：

```cpp
#include  <iostream>
using namespace std;
int f()
{
    int peaches = 1;
    for(int day = 6; day > 0; day --)
        peaches = (peaches+1) /2 ;

    return peaches;
}
int main()
{
    int total_peaches = f();
    cout << total_peaches;
    return 0;
}
```

6.递推算法总结

递推算法的优点是简单易懂、 计算效率高，适用于具有重复性和可分解性的问题。 递推算法适用 于那些可以通过送代求解的问题，如菲波那契数列、 汉诺塔移动次数、 猴子吃桃、 数字三角形（顺推法）、骨牌铺满方格、 蜜蜂路线、吃糖果、 昆虫繁殖、 位数问题、 分苹果、 踩方格等。 在编写递推算法时， 需要注意选择合适的送代方式和终正条件，以确保算法的正确性和效率。