数据结构与算法

# 递归

1. 斐波那契数列
2. N的阶乘
3. 折半查找
4. 汉诺塔问题

//将n个盘子从x借助y移动到z上；

1. 八皇后问题
2. 反向输出

void Print() {

char ch;

int a = scanf\_s("%c", &ch);

if (ch != '#') {

Print();

}

if (ch != '#') {

printf("%c", ch);

}

}

# 字符串

## 模式匹配

### BF算法

时间复杂度：N\*M

### KMP算法

# 动态规划

## 适合解决什么问题？

1. 最值问题；

## 动态规划三要素

1. 重叠子问题，且子问题间必须互相独立；
2. 最优子结构；
3. 状态转移方程；

## 解题套路

1. 明确「状态」；
2. 定义 dp 数组/函数的含义；
3. 明确「选择」
4. 明确 base case

## 背包问题

### 组合问题公式

dp[i] = dp[i] + dp[i-num]

### True、False问题公式

dp[i] = dp[i] | dp[i-num]

### 最大最小问题公式

dp[i] = min(dp[i], dp[i-num] + 1)

或

dp[i] = max(dp[i], dp[i-num] + 1)

* 如果是0-1背包，即数组中的元素不可重复使用，nums放在外循环，target在内循环，且内循环倒序；

**for num in nums:**

**for i in range(target, nums-1, -1):**

* 如果是完全背包，即数组中的元素可重复使用，nums放在外循环，target在内循环。且内循环正序。

**for num in nums:**

**for i in range(nums, target+1):**

* 如果组合问题需考虑元素之间的顺序，需将target放在外循环，将nums放在内循环。

**for i in range(1, target+1):**

**for num in nums:**

编辑距离问题

子序列默认不连续，子数组默认连续

dp[i][j] = max(dp[i-1][j-1]+(A[i-1] == B[j-1]?1:0),dp[i-1][j],dp[i][j-1])

# 深度优先搜索

## 树

1. 基本情况；

* 做事；
* 递归；

前序遍历

### Top Down DFS

* 把值通过参数的形式从上往下传；
* 一般dfs()本身不返回值；

### Bottom Up DFS

* 把值从下往上传；
* 当前递归层利用子问题传上来的值计算当前层的新值并返回；
* 一定会有返回值；

**一般流程**

1. 基本情况；
2. 向子问题要答案；
3. 利用子问题答案构建当前问题答案；
4. 返回答案给父问题。

## 图

## 数组和矩阵