分治 + 回溯 + 递归 + 动态规划

(将复杂的问题，分解成各种子问题，同时寻找它的重复性)

Java 代码模板 递归

```java
// Java
public void recur(int level, int param) {

  // terminator
  if (level > MAX_LEVEL) {
    // process result
    return;
  }

  // process current logic
  process(level, param);

  // drill down
  recur( level: level + 1, newParam);

  // restore current status

}
```

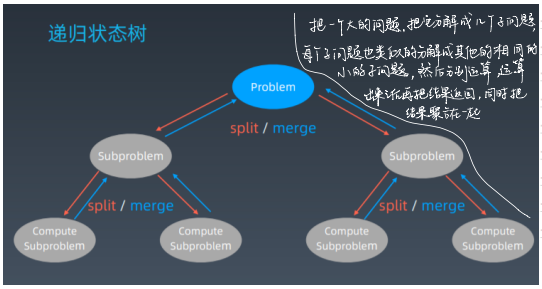
递归终止条件

处理当前层逻辑

下探/递归到下一层

恢复当前层状态

(简单变量不需要此步骤)




```java
Java

private static int divide_conquer(Problem problem, ) {

  #recursion terminator
  if (problem == NULL) {
    int res = process_last_result();
    return res;
  }

  #prepare data
  subProblems = split_problem(problem)

  #conquer subproblems
  res0 = divide_conquer(subProblems[0])
  res1 = divide_conquer(subProblems[1])

  #process and generate the final result
  result = process_result(res0, res1);

  return result;
  #revert the current level states
}
```

分治(递归的一种特殊形式)

递归终止条件

拆分子问题

调子问题递归的函数 = drill down

合并结果

(有时能需要)恢复当前层状态

动态规划 Dynamic Programming

1. Wiki 定义：
   https://en.wikipedia.org/wiki/Dynamic_programming

2. "Simplifying a complicated problem by breaking it down into simpler sub-problems"
   (in a recursive manner) (用一种递归的方式)

3. Divide & Conquer + Optimal substructure
   分治 + 最优子结构

DP：求最优解/最大值/最小的方式

将一个复杂的问题，把它分解成简单的子问题

只存最优的状态

关键点：

动态规划 (本质：动态递推)

→ 复杂度更低、更加有效

动态规划和递归或者分治没有本质上的区别 (关键看有无最优的子结构)

没有最优子结构

↓说明

所有的子问题你都需要计算一遍，

同时把最后的结果结合在一起

↘一般称之为"分治"

共性：找到重复子问题

差异性：最优子结构、

中途可以淘汰次优解

(必须)