2021/3/21

递归模板

Void fun(level, param){

// recursion terminator

If(level > MAX\_LEVEL){

//process result

Return;

}

// process current logic

Process(level,param);

// drill down

fun(level+1,param)

// reverse current level state if needed

}

分治 模板1

Int devide\_conquer(level, param){

// recursion terminator

If(problem == nullptr){

process result

Return return\_result;

}

//process current problem

Subproblems = split\_problem(problem, data)

Subresult1 = devide\_conquer(subproblem[0], p1)

Subresult2 = devide\_conquer(subproblem[1], p1)

Subresult3 = devide\_conquer(subproblem[2], p1)

...

//merge

Result = process\_result(sub\_result1, sub\_result2, subresult3)

//revert the current level status if needed

Return 0;

}

分治 模板2

Int devide\_conquer(problem, param){

// recursion terminator

If(problem == nullptr){

process result

Return return\_result;

}

//prepare data

Data = prepare\_data(problem)

Subproblems = split\_problem(problem, data)

//conquer subproblems

Subresult1 = devide\_conquer(subproblem[0], p1)

Subresult2 = devide\_conquer(subproblem[1], p1)

Subresult3 = devide\_conquer(subproblem[2], p1)

...

//process and generate the final result

Result = process\_result(sub\_result1, sub\_result2, subresult3,...)

//revert the current level status if needed

Return 0;

}