回溯法复习

Hushrushing.2021.05.03

**回溯法**

* 回溯法的本质是**深度优先搜索**；
* 用回溯法解决的问题需要具有若干**特点**：
* 问题的解由**向量表示**；
* 所有的解可以**构成解空间树**；
* 深度优先遍历解空间树，如果某一结点肯定不符合问题约束，则**跳过**以该结点为根结点的子树。
* 注意：同一个问题可以有多种表示，简单的表示方法可以获得更好的效果。

回溯法的具体步骤

1. **定义问题解空间；**
2. **确定解空间树；**
3. **深度优先方式搜索解空间，并在不符合约束的结点剪枝。**

回溯法解题的显著特征

* 在任何时刻算法只保存从根结点到当前结点的路径，所需计算空间通常为。（最长路径为）
* 一般情况下，可以用递归实现回溯[[1]](#footnote-1)：

void backtrack(int t)

{

    if(t>n)//n是递归深度,t>n时已经到叶子结点

        output(x)；

    else

        for (int i = f(n, t); i <= g(n, t); i++)

        {

            x[t] = h(i);

            if (constraint(t) && bound(t))

                backtrack(t + 1);//退出递归搜索

        }

}

* 也可以用非递归的方式实现。

void iterativeBacktrack()

{

    int t=1;

    while(t>0)

    {

        if(f(n,t)<=g(n,t))

        for(int i=f(n,t);i<=g(n,t);i++)

        {

            x[t]=h(i);

            if(constraint(t)&&bound(t))

            {

                if(solution(t))

                output(t);

                else

                t++;

            }

        }

        else t--;

    }

}

例1装载问题

* 1. **题目描述**

有一批共n个集装箱要装上2艘载重量分别为c1和c2的轮船，其中集装箱i的重量为wi，且。装载问题要求确定是否有一个合理的装载方案可将这n个集装箱装上这2艘轮船[[2]](#footnote-2)。如果有，找出一种装载方案。

* 1. **题目分析**

装载策略是**（1）首先将第一艘轮船装满，（2）再将剩余集装箱装上第二艘轮船**[[3]](#footnote-3)。故只需选出装到第一艘中的集装箱即可。装载问题等价于以下**特殊0-1背包**问题：

都是0-1变量，目标就是选出能装上第一艘船的最大集装箱集合。

可用图1.1表示解空间子集树

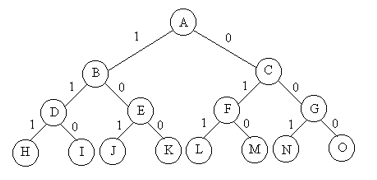


图1.1 解空间子集树示意图

其中0表示不选，1表示选中。算法过程即深度优先搜索，但求得总重量有标记。用cw表示当前结点处装载总重量，bestw表示最大重量，在递归的过程中不断优化，但cw>船的限重时[[4]](#footnote-4)就直接跳过该结点。代码如图1.2所示。

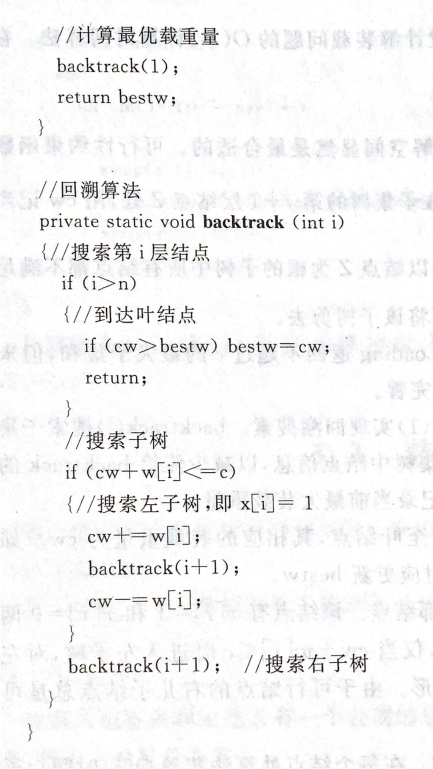
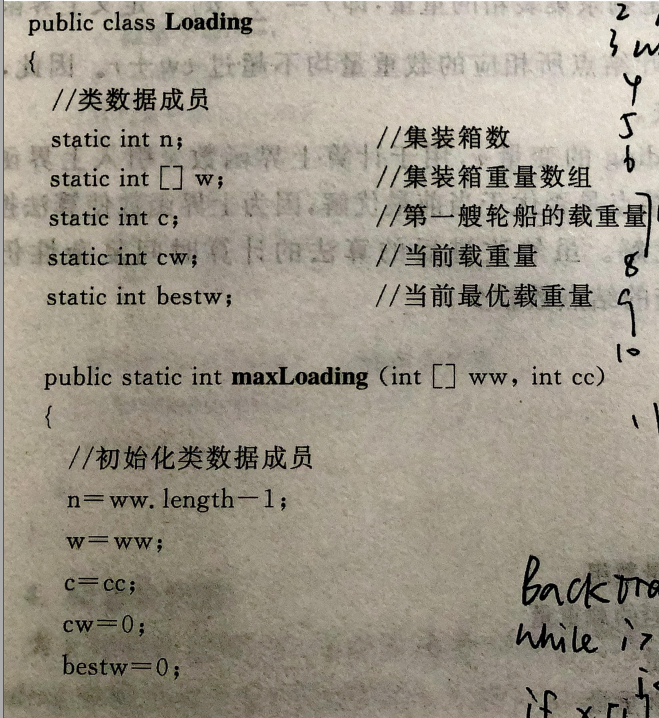
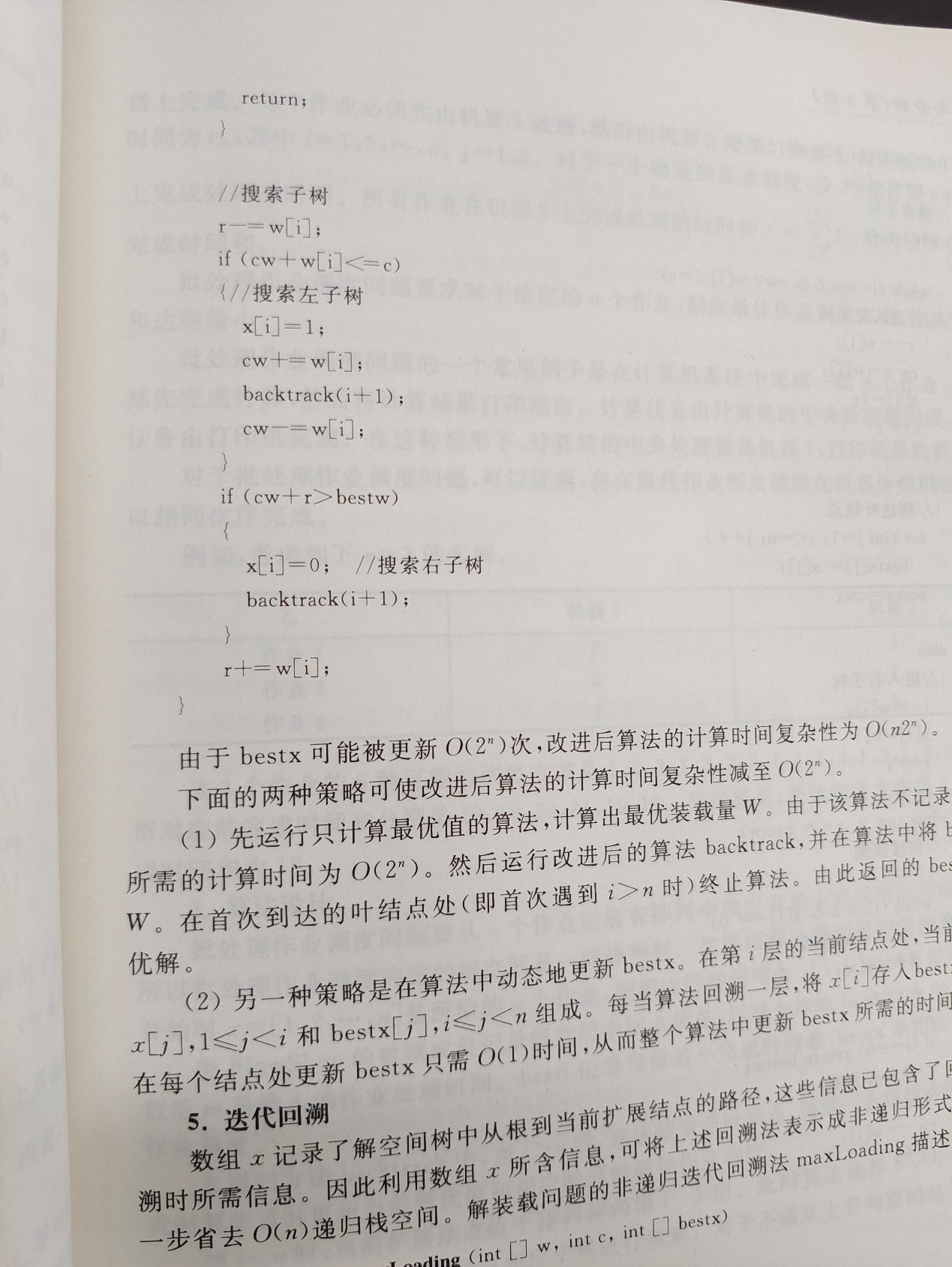
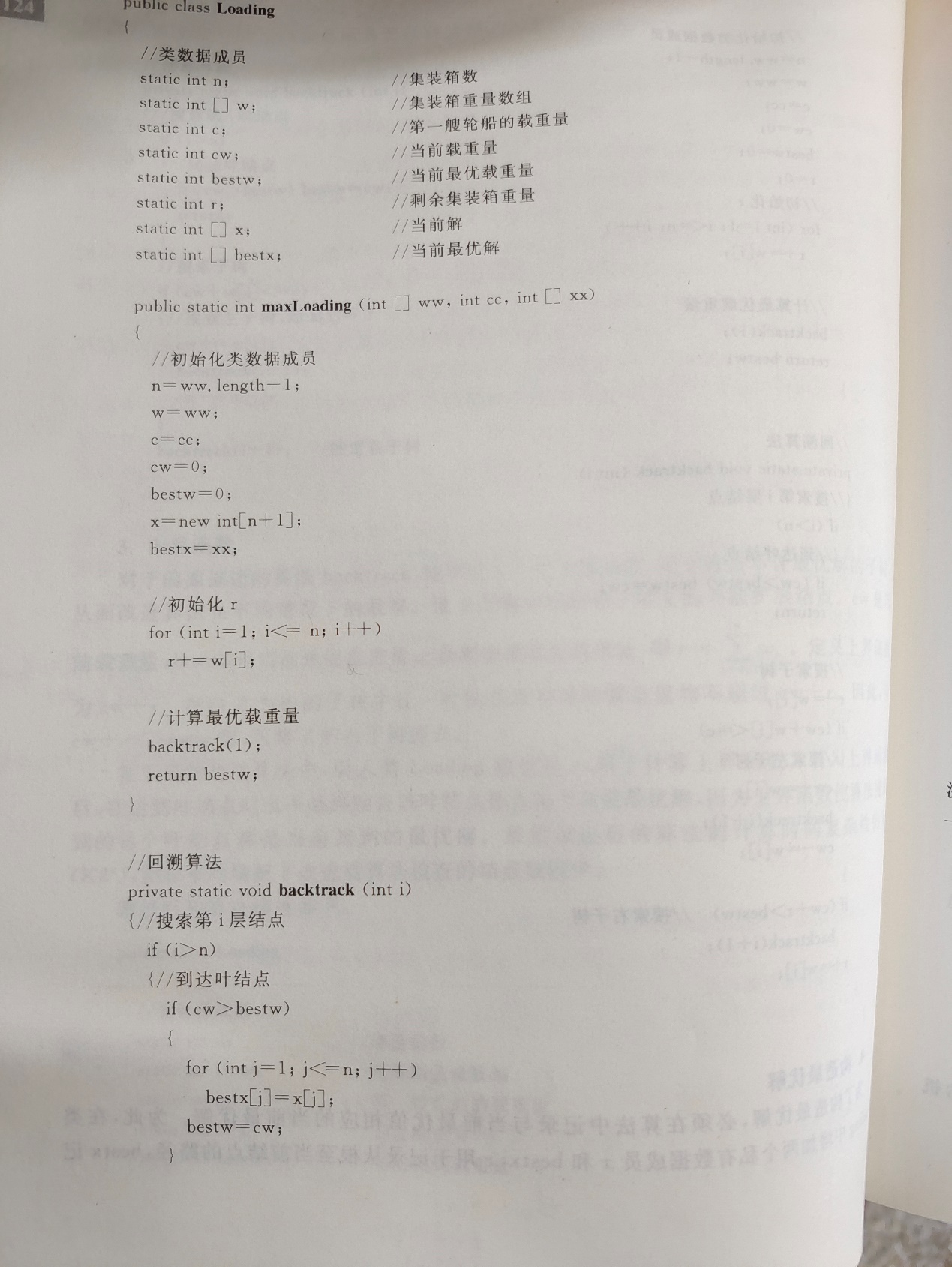
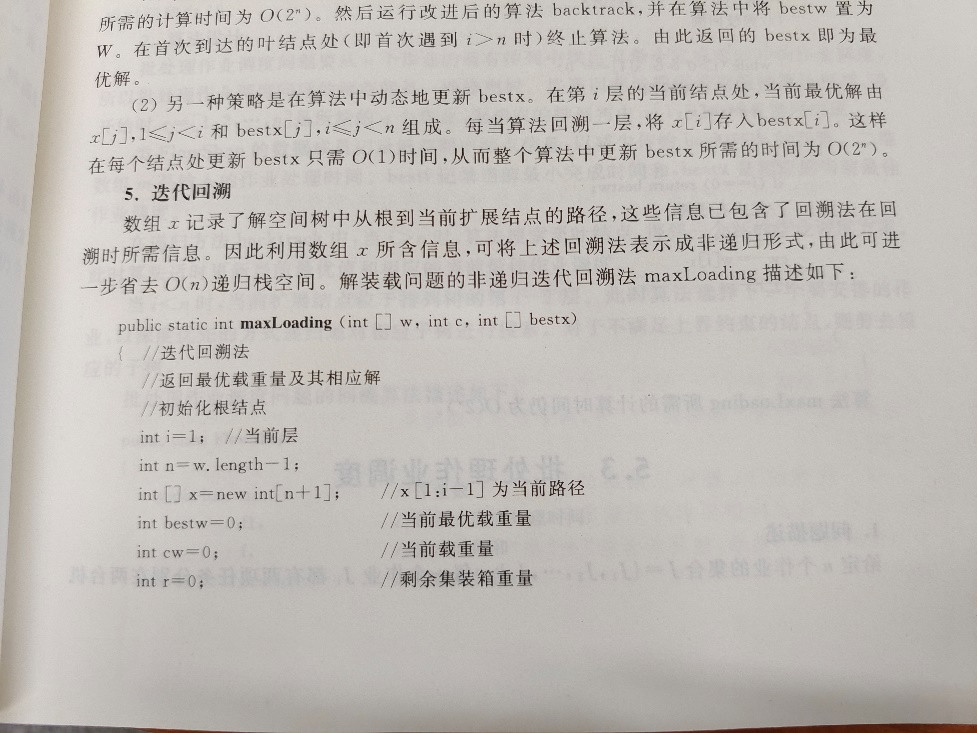


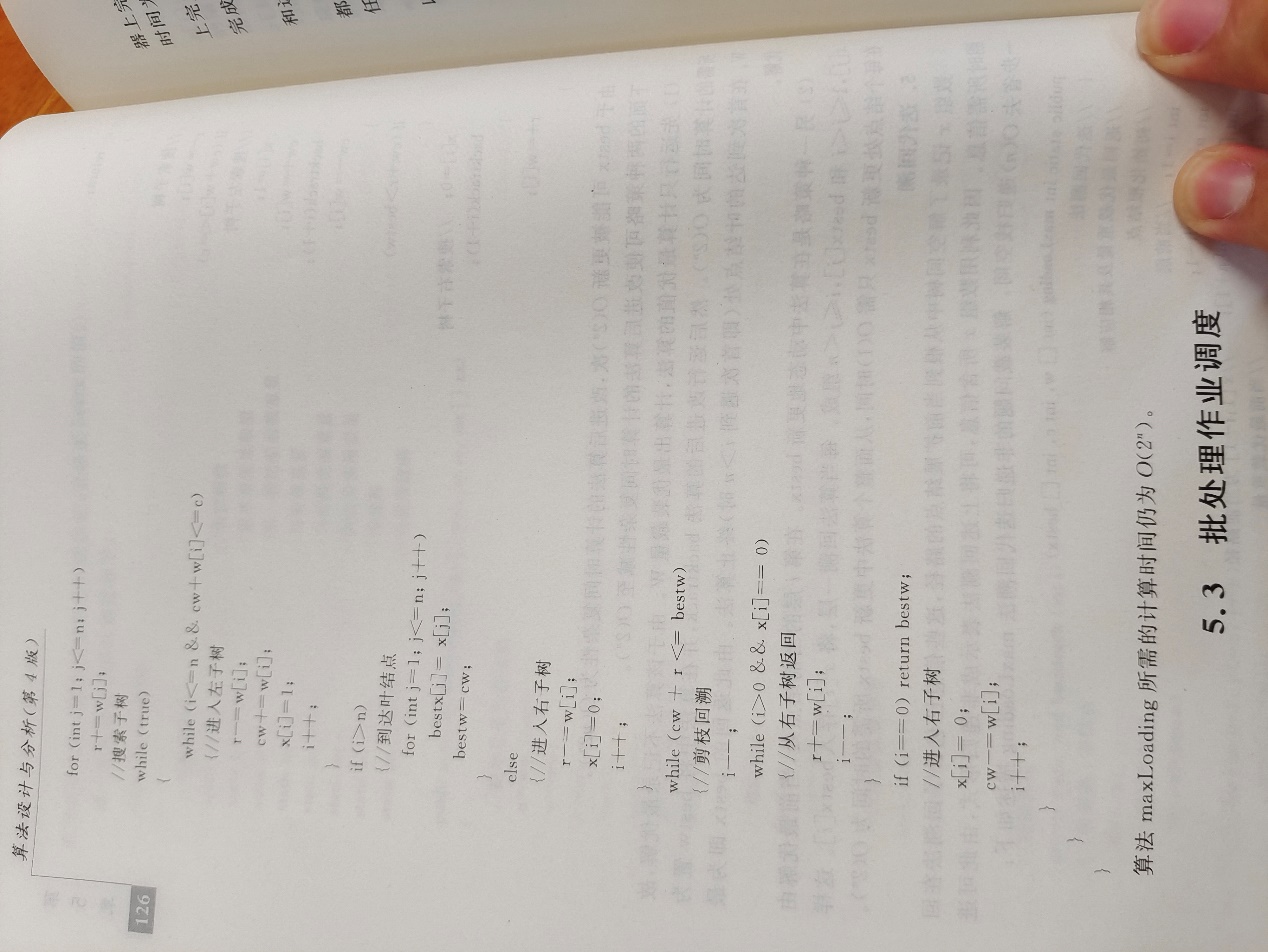
图1.2 代码示意

还可以定义上界函数，原理是在某一结点处，设剩下可选的集装箱总重量为r，若cw+rbestw，则该结点以下不可能是最优解。算法结束后应验证剩余集装箱能否装上第二艘船。

为了构造解，可增加变量或成员记录当前搜索路径。代码如下：

也可以不用递归。





1. 这个过程最好用实例解释。 [↑](#footnote-ref-1)
2. 注意题目说的是将所有集装箱装上船。 [↑](#footnote-ref-2)
3. 这个装载问题只有重量限制。 [↑](#footnote-ref-3)
4. 如果将左孩子结点设为1而右孩子结点设为0，则进入右孩子结点时不需要检查。 [↑](#footnote-ref-4)