最大int: 0x7fffffff 最小int: 0x80000000

二分法:

二分法一般用于题目中有单调性的情况，比如说最传统的有序序列中的查找，下标与值的关系就是单调递增的。除此之外，题目中的单调性也许十分隐蔽，如leetcode 410。

410 363

二叉树:

递归时候可通过参数来存值

LCA 最近公共祖先

F(x) = x结点为根，节点到子树最远的距离

以x为最近公共祖先的最大path, maxdeep[x->left] + maxdeep[x->right]

有序序列建立平衡二叉树：找中点然后递归建立 (构建平衡二叉树: 先排序再这么建立？)

树:

1. In computer science, a trie, also called digital tree, radix tree or prefix tree, is a kind of search tree—an ordered tree data structure used to store a dynamic set or associative array where the keys are usually strings. 字典树(快速搜索单词)：

前缀树的构造过程，通过不断插入新的字符串来丰富这棵26叉树。**强调注意这里是26叉树**，因为每一个英文字符串中下一个字母都只能是a-z中的一种可能。

<https://blog.csdn.net/qq_33297776/article/details/82315859> leetcode 208

前缀树和后缀树: <https://blog.csdn.net/u013949069/article/details/78056102>

1. 线段树:

一维坐标上有若干条线段，起点st，终点ed，0<=st<ed<=maxint

问一个点被多少条线段覆盖

操作insert(int st, int ed);

查询query(int x)

多次插入与查询

深度优先搜索：

框架 需要原料和边界(index)作为参数 leetcode 46 77 39

往往需要确认index代表的含义

用二进制来未压缩，for来实现递归的效果 leetcode 216

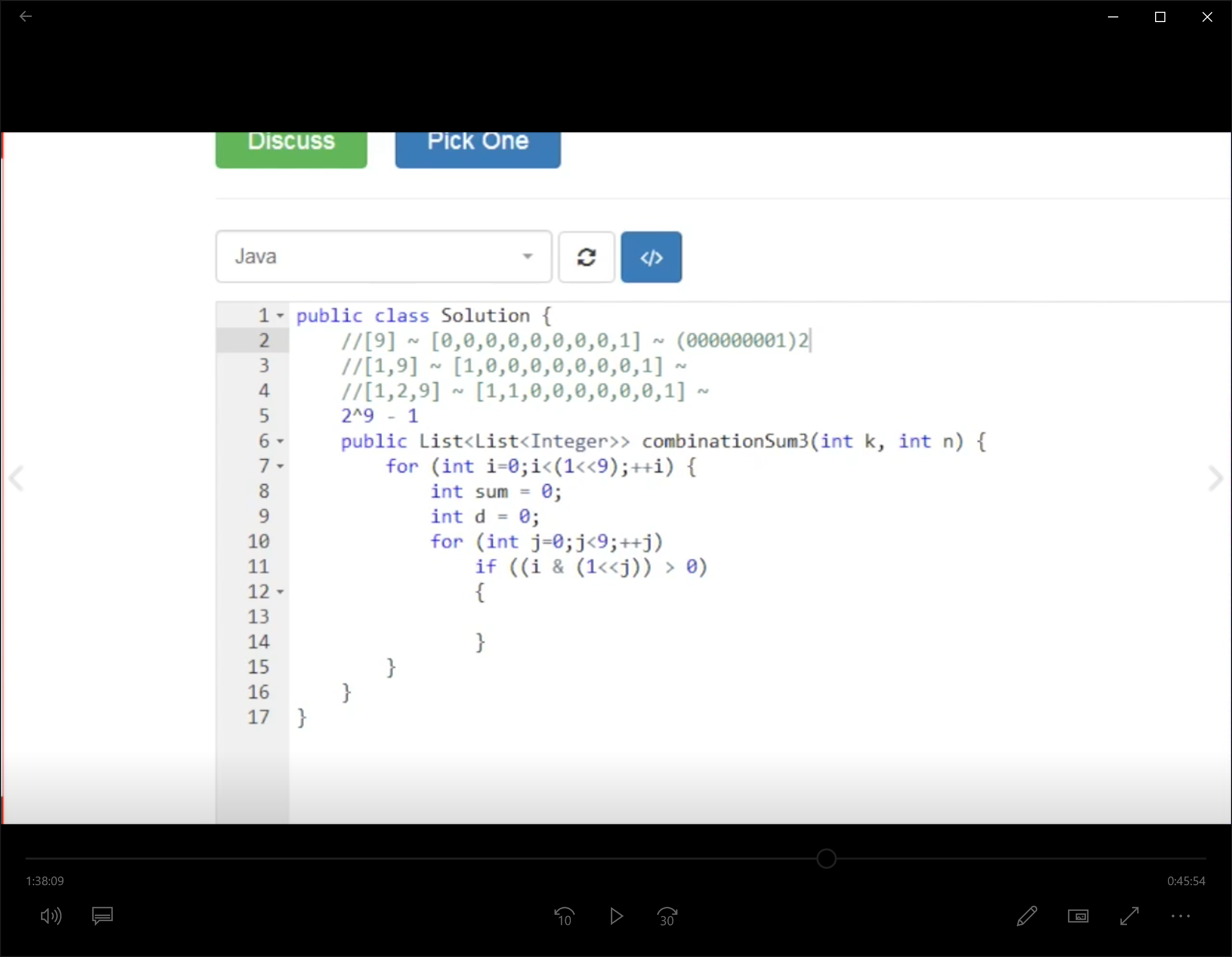


图:



并查集 leetcode 399 130 200

Prim与djikstra的区别：

Prim每次选点之后都只把过去的结果与新增的边作比较，Dijkstra是用新的点与过去的点和所连的边的合作比较

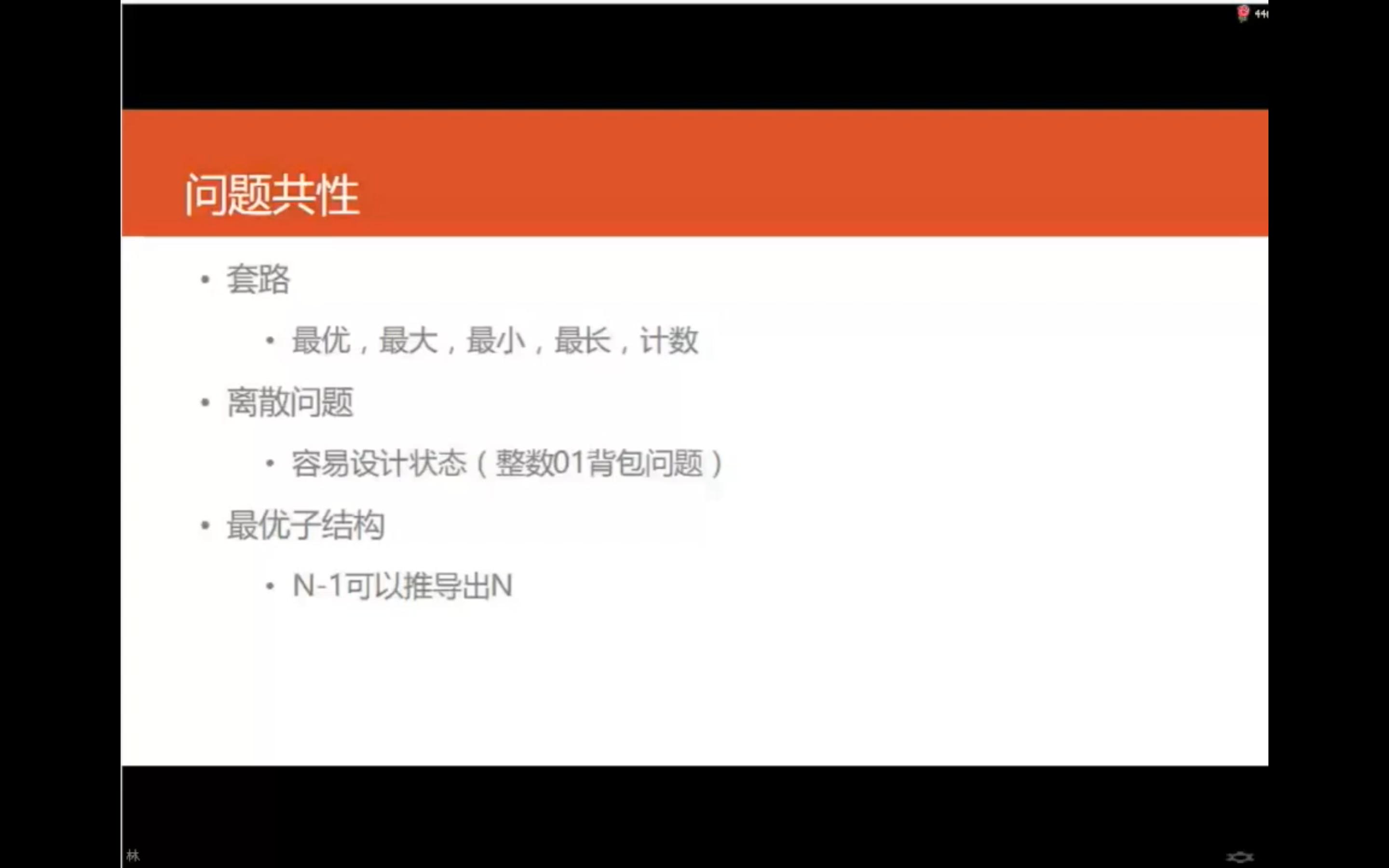
检查图是否有回路：

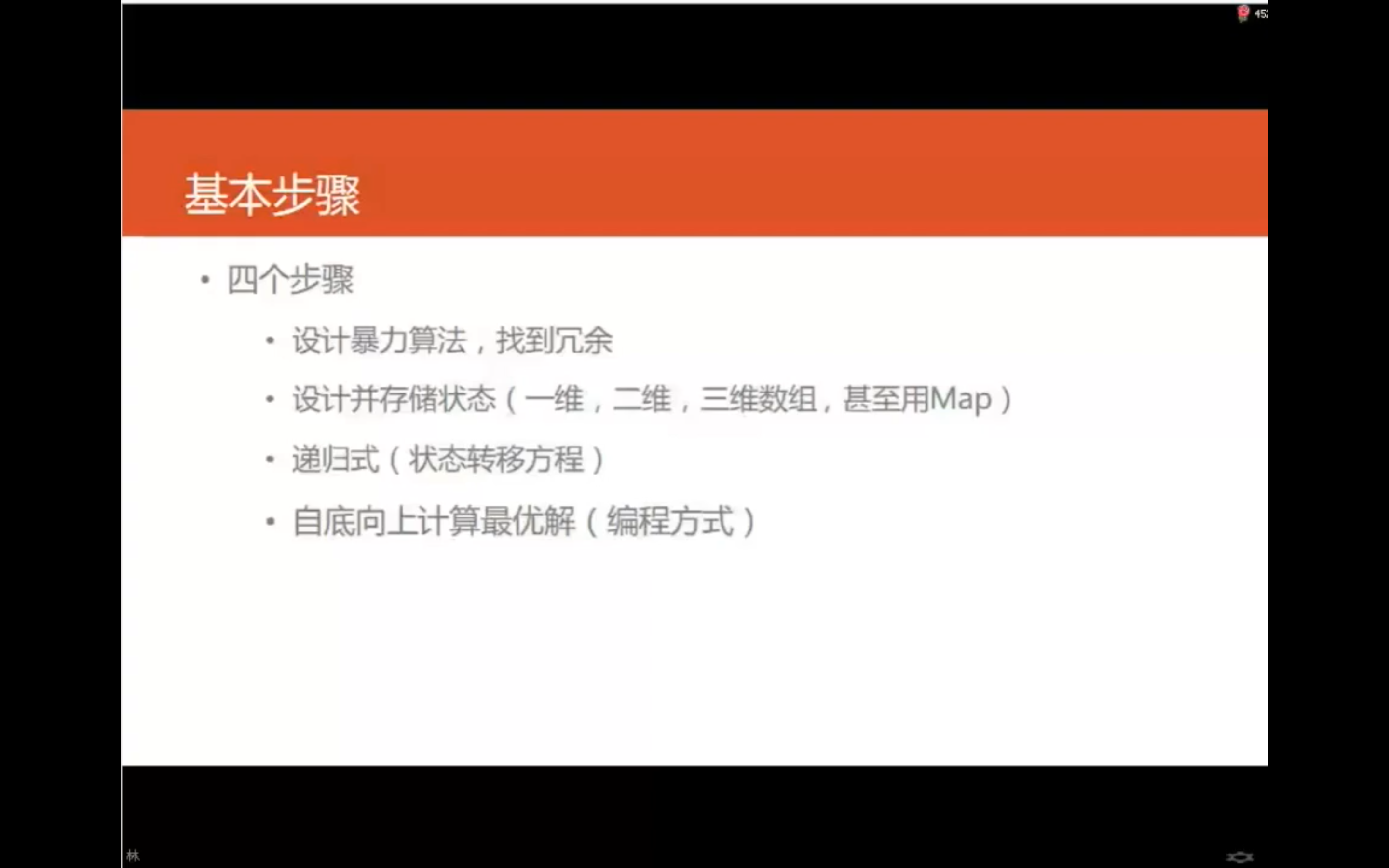
方法1：并查集

动态规划:

动态规划就是没有后效性的递归，用空间换时间，将算出过的子问题解保存下来，下次如果碰到这个子问题直接调出答案

动态规划特征：





节省空间可用滚动数组

01背包问题是取或者不取的问题，index去原数或加1，但是用深度优先搜索得用for，不好改成动态规划。如果保存数组是二维，那么尽量参加计算的数组的下标有一边在一起，不然很难改写成动态规划。如果是这种情况，就直接想动态规划，自底向上比较好，例如leetcode 322 (<https://blog.csdn.net/h2453532874/article/details/89139309> 最好写成一维)。

回溯法：

经典思路：保存回溯结果需要用两个数组，一个存最终结果，一个存路径。函数调用是首先放一个元素，调用之后的，然后把元素拿出来，再调用之后的.

