



序的应用

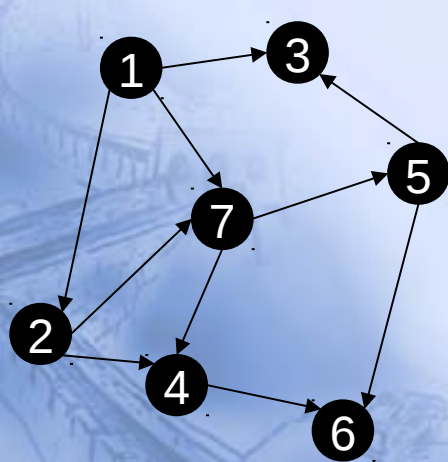
长沙市雅礼中学 龙凡

基本的序

序是数据之间隐藏的一种基本关系：

4 12 31 24 43 25 34 \longrightarrow 4 12 24 25 31 34 43

大小序 \approx \top



\longrightarrow 1 2 4 6 7 5 3 DFS 序

\longrightarrow 1 2 4 7 5 3 6 拓扑序



序的应用

- 使得一个问题得到直接解决（大多是交互式问题）
- 应用序，挖掘题目的深层性质，使得问题得到转化。



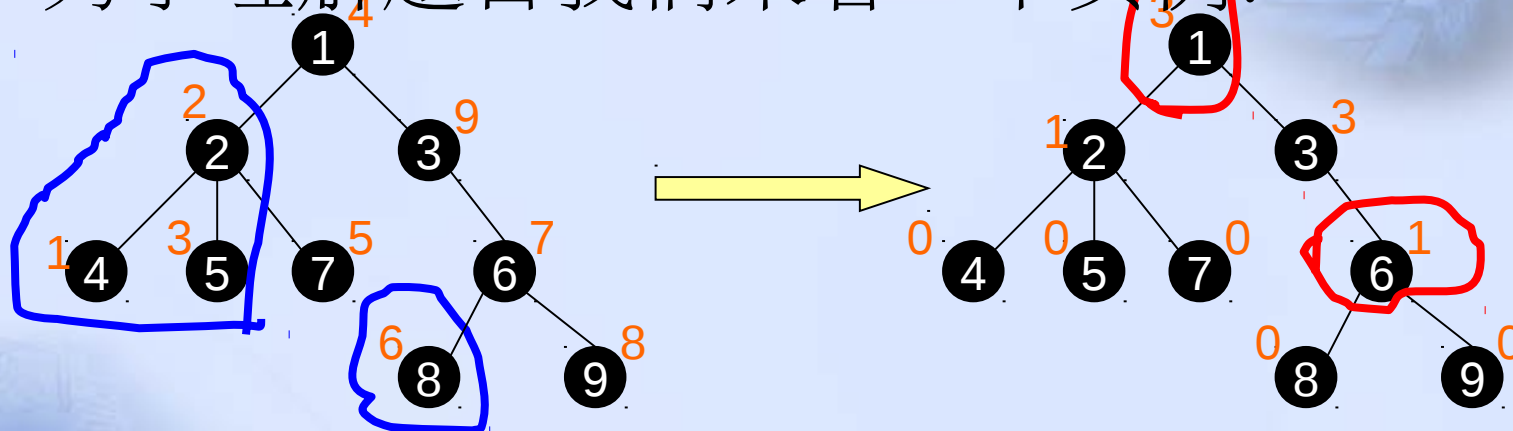
树的构造

问题描述：一棵含有 n 个节点的树，所有的节点依次编号为 $1, 2, 3, \dots, n$ ，每个节点 i 有一个权值 $s(i)$ ，也分别是 $1, 2, 3, \dots, n$ ，并且各不相同。对于编号为 v 的节点，定义 $t(v)$ 为 v 的后代中所有权值小于 $s(v)$ 的节点个数。

现在给出这棵树及 $t(1), t(2), \dots, t(n)$ ，请你求出这棵树每个节点的权值。

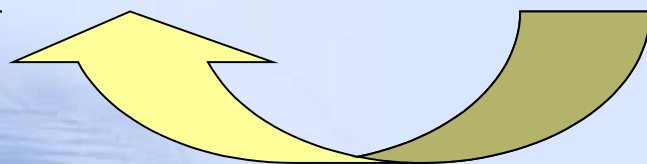
树的构造

为了理解题目我们来看一个实例：



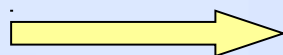
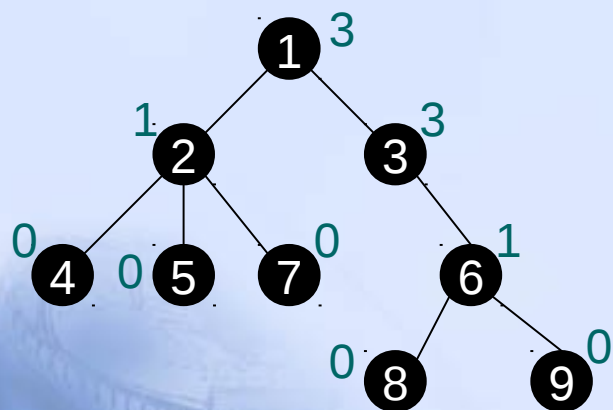
构造 S

已知 T



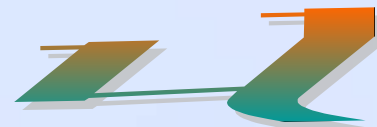
树的构造

我们来考虑这个树的 DFS 序列:



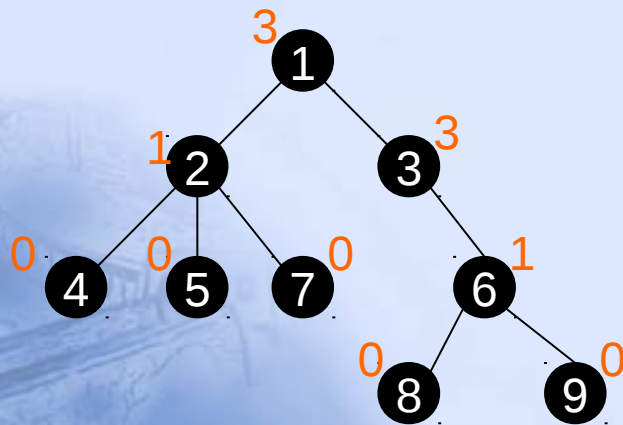
1(3)2(1)4(0)5(0)7(0)3(3)6(1)8(0)9(0)

DFS 序列的重要性质



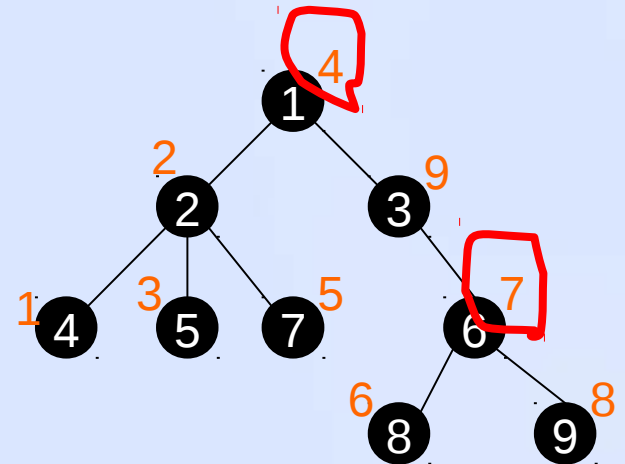
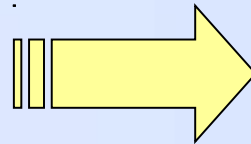
树的构造

由于权值分别是 $1, 2, 3, \dots, n$ 。我们不妨认为从左到右有 N 个格子，如果从左数第 i 个格子填入了节点 J ，则 $S(j)=i$ 。



左数第
位置

左数第 7 个
位置

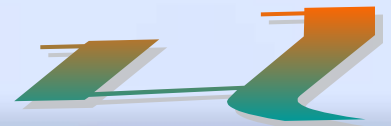
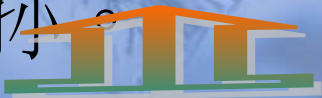


一组可行解

树的构造

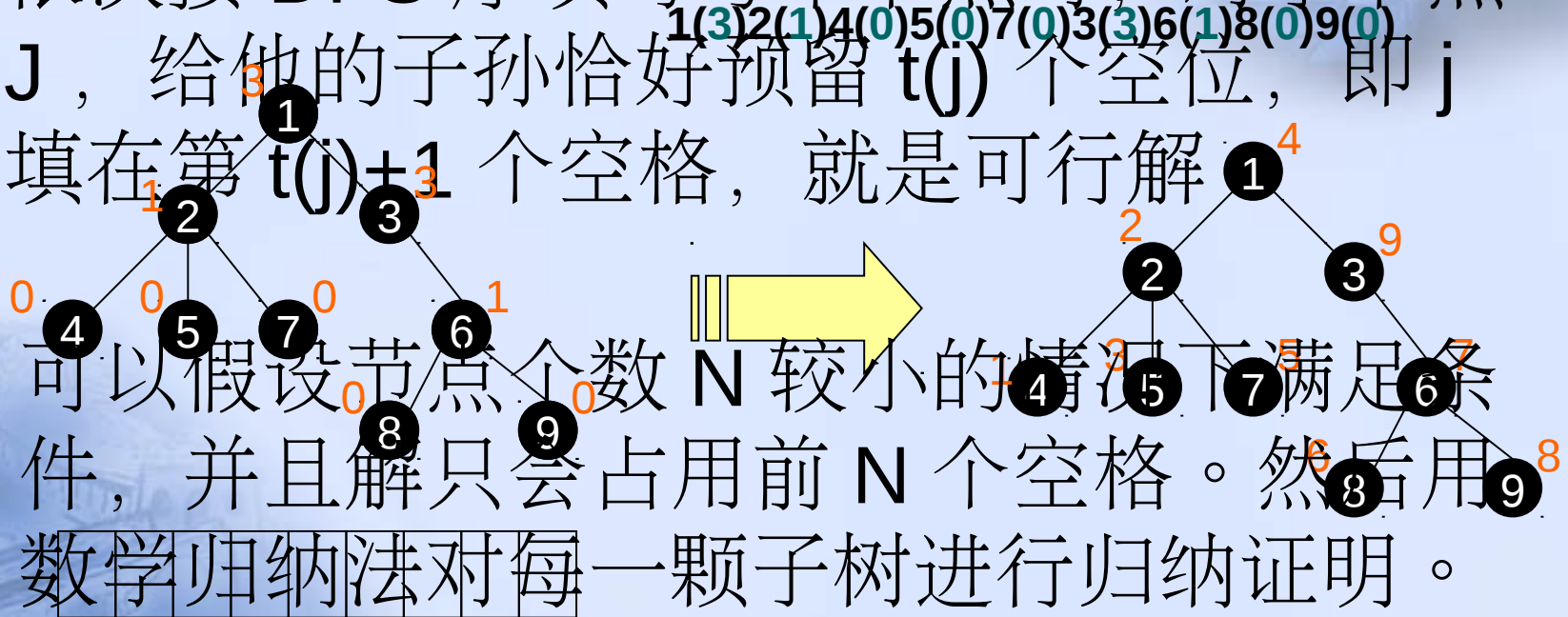
- 不能“看漏填”，的时候要满足具体要求
- 每个格子的左边，都恰好有 $t(i)$ 个格子填入了自己的子孙。
- 不能超出 $1 \dots n$ 的边界范围。

如果我们按照 **DFS** 的顺序，依次填写节点。对于每个节点 j 的左边，则必须预留至少 $t(j)$ 个空格给权值比他小的子孙。



树的构造

依次按 DFS 序填写每个节点时，对于节点 J ，给他的子孙恰好预留 $t(j)$ 个空位，即 j 填在第 $t(j)+1$ 个空格，就是可行解



树的构造

看看转化后的问题:

- 已知一个一维线形结构
- 最开始所有位置为空。根据 **DFS** 序列，每次插入一个元素 j ，到第 $t(j)+1$ 个空位置
- 求出最终状态
- 借助线段树或树状数组等数据结构可以将问题在 $O(N \log N)$ 时间复杂度内解决，空间复杂度为 $O(N)$

小结

- 通过对题目特点的分析，借助 **DFS** 序列的性质，对原问题进行转化。
- 合理的使用数据结构，最终完整解决问题。

士兵排队

问题描述: N 个士兵在进行队列训练, 从左至右有 M 个位置。每次将军可以下达一个命令, 表示为 $\text{Goto}(L, S)$

1. 若队列 L 位置上为空, 则士兵 S 站在 L 上。
2. 若是把 L 位置及其右方的, 一段士兵 S 向右推一格, 并执行 $\text{Goto}(L, S)$, 然后 S 站在 L 上。

将军依次下达 N 个命令, 每个士兵被下达命令一次且仅一次。要你求出最后队列的状态。(有可能在命令执行过程中, 士兵站的位置标号超过 M)

士兵排队

一个简单的例子

Goto(4,1)

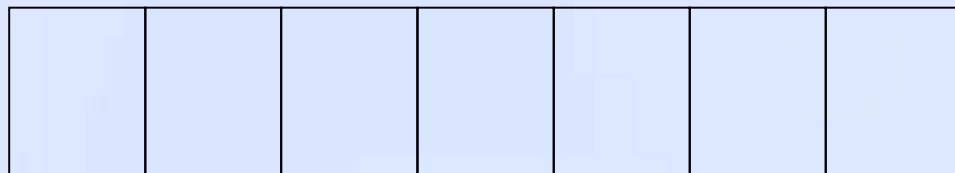
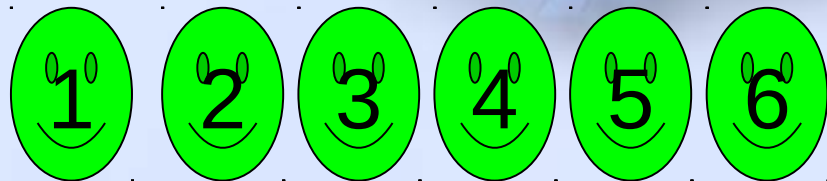
Goto(4,2)

Goto(5,3)

Goto(2,4)

Goto(4,5)

Goto(3,6)



N=6 M=5

士兵排队

我们来进行一下初步分析：

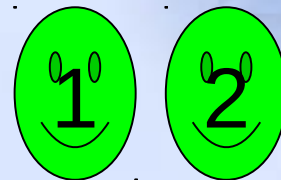
- 直接模拟的最坏时间复杂度为 $O(N^2)$ ，效率十分低下。
- 使用平衡二叉树，可以得到一个 $O(N \log(N+M))$ 的算法。但平衡二叉树时间复杂度常数系数比较大，而且较难实现。
- 不妨抛开纯粹模拟的思路，另辟蹊径。

士兵排队

先来看最基本的情况：

Goto(2,1)

Goto(2,2)

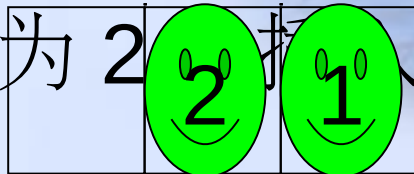


可见：如何高效处理插入带来的连锁移动是本题的关键！



士兵排队

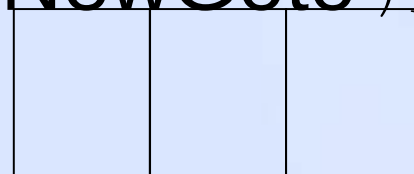
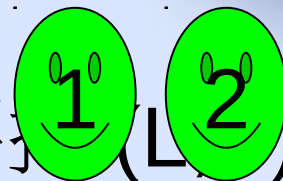
- 注意到上面的例子中 1 因为 2 挡住了而向右移动了一格。


- 我们要避免连锁移动，就是希望通过一个规则，使得士兵 1 能够直接插入到 3 号位置。我们可以先插入士兵 2 而不是士兵 1，然后再将士兵 1 插入到第 2 个空位置中。
- 具体地说，定义： `newgoto(L,S)` 命令，将 S 士兵插入到第 L 个空位置中。

士兵排队

NewGoto(2,2)

事实上原 Goto 序列只要把 (L,R) 数
对合理交换位置, 就和 NewGoto 序
列等价

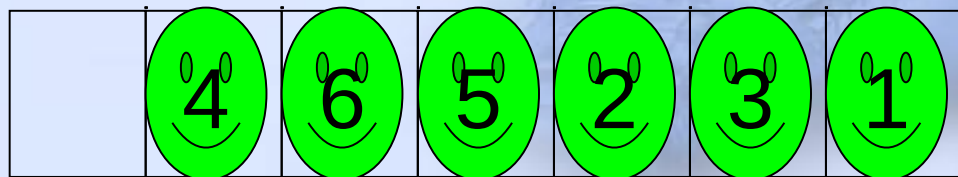


Goto(2,1)

Goto(2,2)

士兵排队

复杂一点的情况:



Goto(4,1)

NewGoto(4,5)

Goto(4,2)

NewGoto(5,3)

Goto(5,3)

NewGoto(4,2)

Goto(2,4)

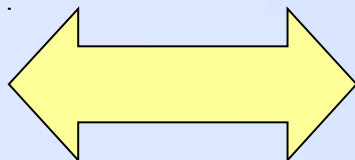
NewGoto(4,1)

Goto(4,5)

NewGoto(3,6)


Goto(3,6)

NewGoto(2,4)



士兵排队

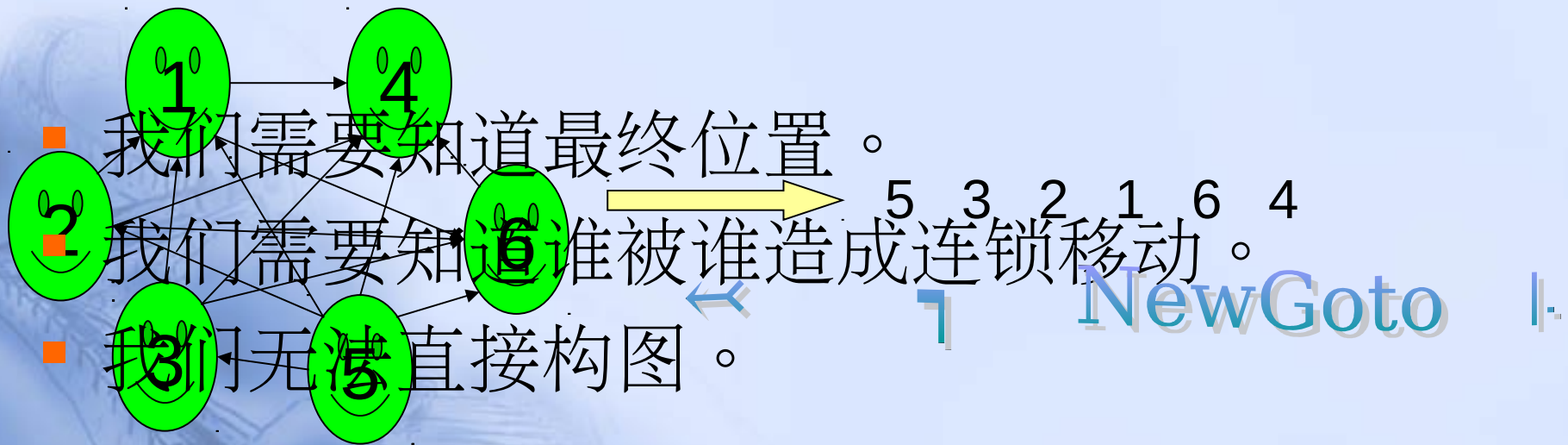
- 如果我们可以通过将 Goto 序列的 (L,S) 数对, 高效合理的改变顺序, 转化为 NewGoto 序列, 则模拟 NewGoto 命令后转化成一维线性填数问题。


 Goto(LB,B)
 NewGoto(LA,A) 的原则: 第 LA 个空位置

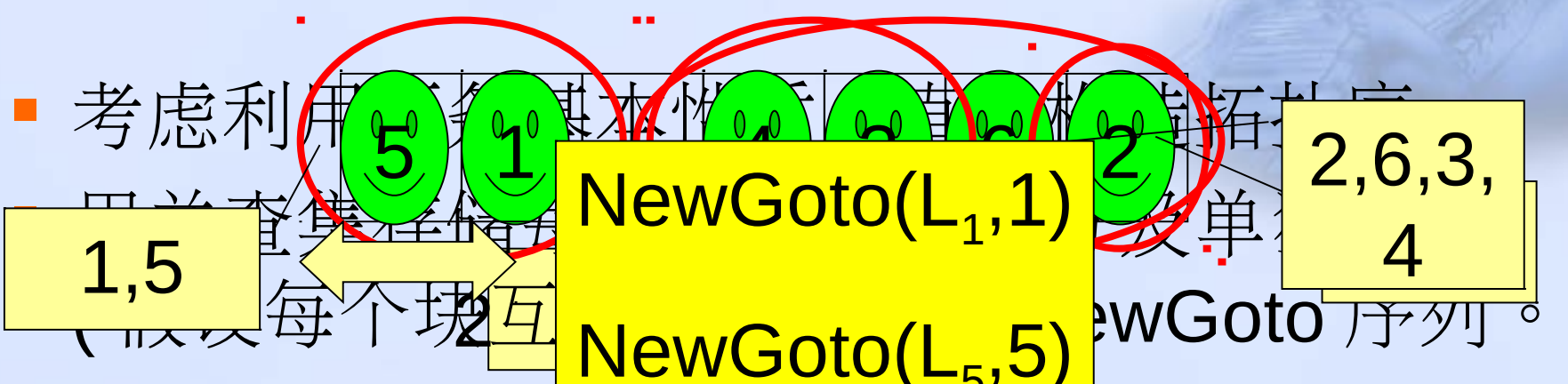
- 如果 A 因 B 插入而被连锁移动。则和 A,B 有关的两条 NewGoto 命令, B 要在 A 之前。
- 如果 A,B 没有关联, 而 A 最终位置在 B 之前, 则 NewGoto 序列中, B 要在 A 之前。

士兵排队

如果构造一个图，与 **A** 相关的 **NewGoto** 命令要在与 **B** 相关的之前，则 **A,B** 之间连一条边 **A->B**，那么我们就是要获得这个图拓扑序。

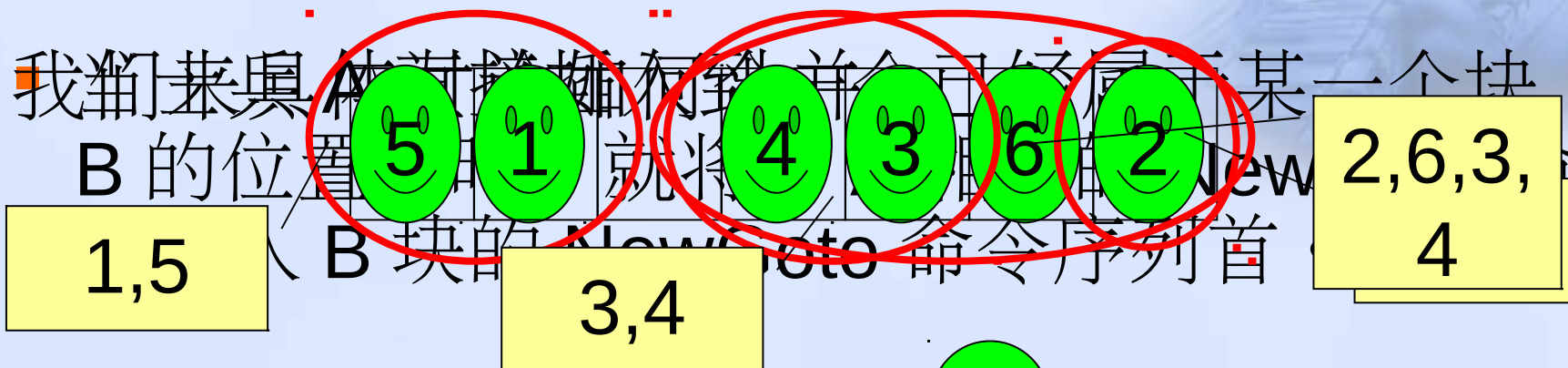


士兵排队

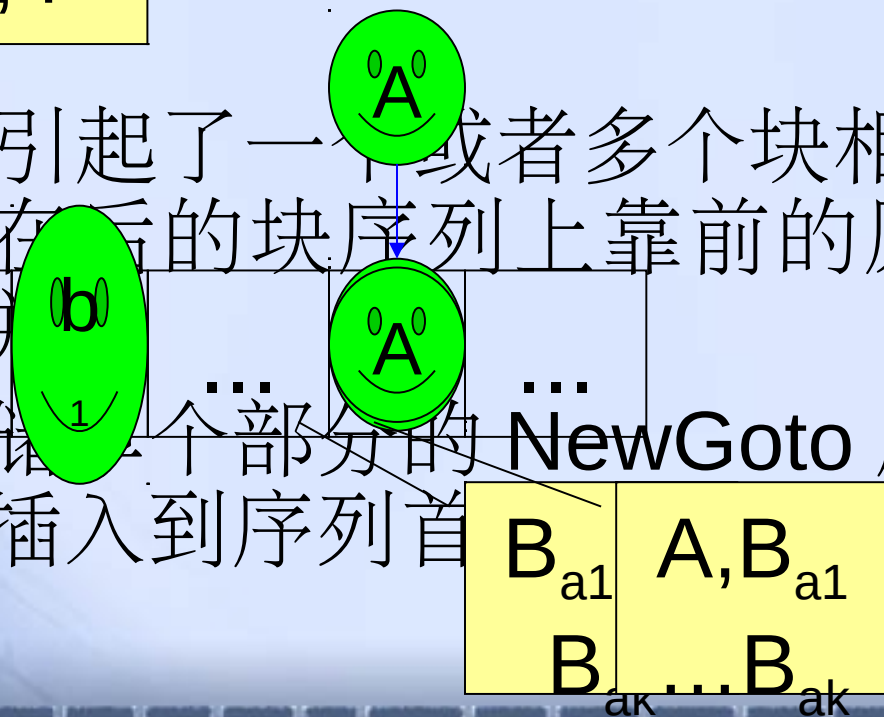
- 考虑利用每个块的基本性质，构造 NewGoto 序列。

The diagram illustrates the construction of a NewGoto sequence. It shows a sequence of soldiers (5, 1, 4, 3, 2) with red arcs indicating connections between them. Yellow boxes contain the following text: '1,5', 'NewGoto(L₁,1)', 'NewGoto(L₅,5)', and '2,6,3,4'.
- 当两个块因为插入而安合开时，顺便将两个块的 NewGoto 序列合并。
- 最后将所有未合并的部分的序列，根据位置在后的块序列上靠前的原则，合并完整的 NewGoto 序列。

士兵排队



- 当士兵 A 的插入引起了一个或者多个块相连时，则根据位置在后的块序列上靠前的原则来对他们进行合并
- 用一个链表来存储各个部分的 NewGoto 序列，因为他只涉及插入到序列首并两个操作



士兵排队

具体实现的例子:

Goto(4,1)

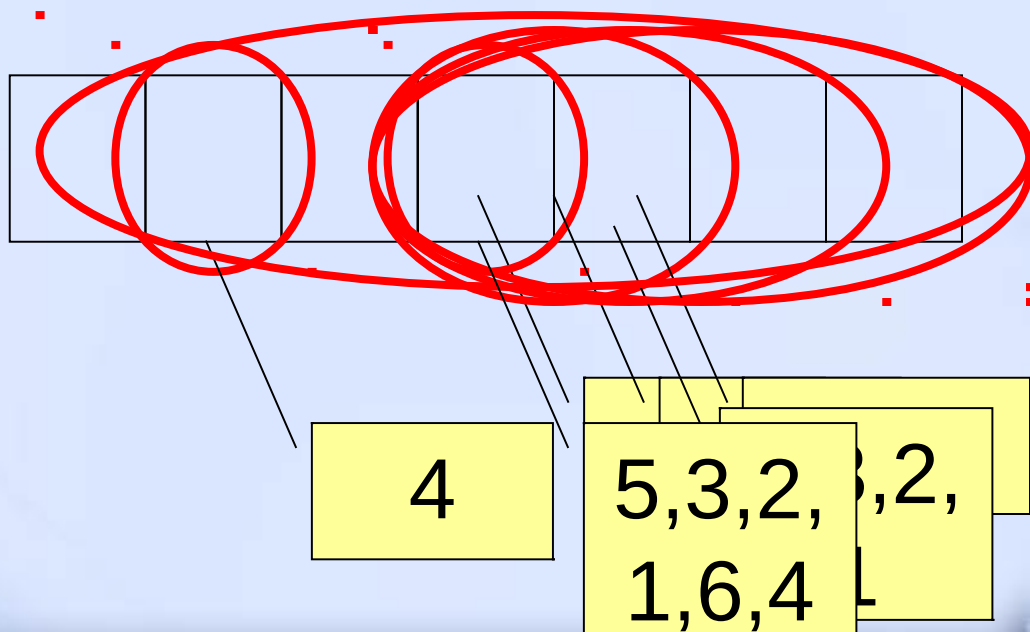
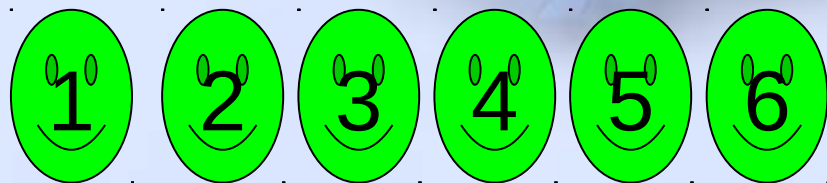
Goto(4,2)

Goto(5,3)

Goto(2,4)

Goto(4,5)

Goto(3,6)



士兵排队

- 现在来整理整个算法：
根据 **Goto** 序列构造 **NewGoto** 序列。转化成前一题最终转化成的一维线性填数问题。
- 使用线段树等工具在 $O(N \log (N+M))$ 时间内解决转化后问题。
- 总时间复杂度 $O(N \log (N+M))$ ，空间复杂度 $O(N+M)$ 。

总结

- 通过适当的分析，应用不同的序，将两个截然不同的问题转化成了同一个问题。
- 提示我们在平时做题时要充分挖掘题目的本质。
- 同时大胆的尝试，去实现自己的想法。

