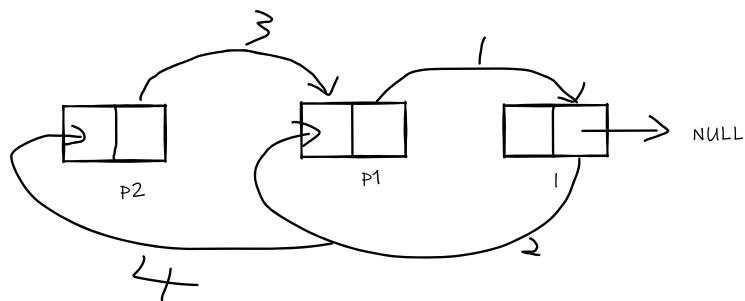
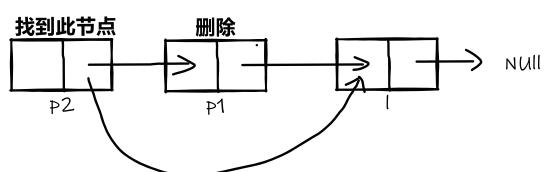


单链表

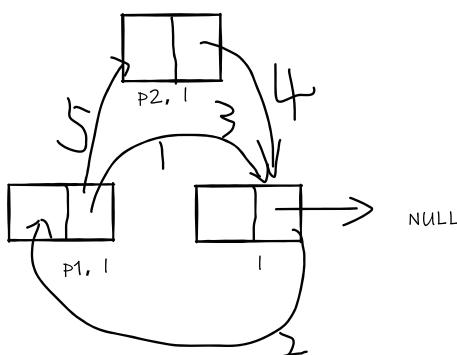
头插法



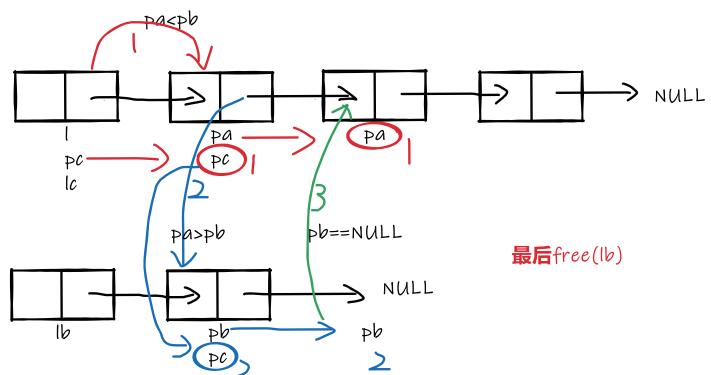
删除



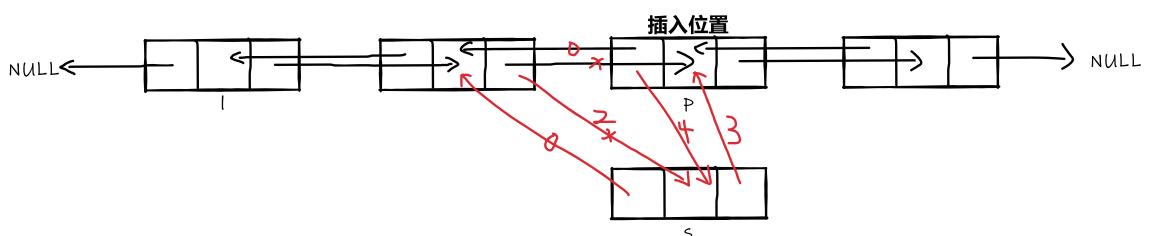
尾插法



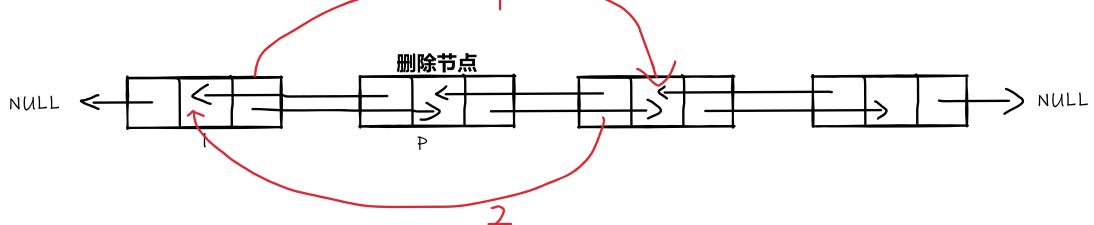
非递减单链表归并



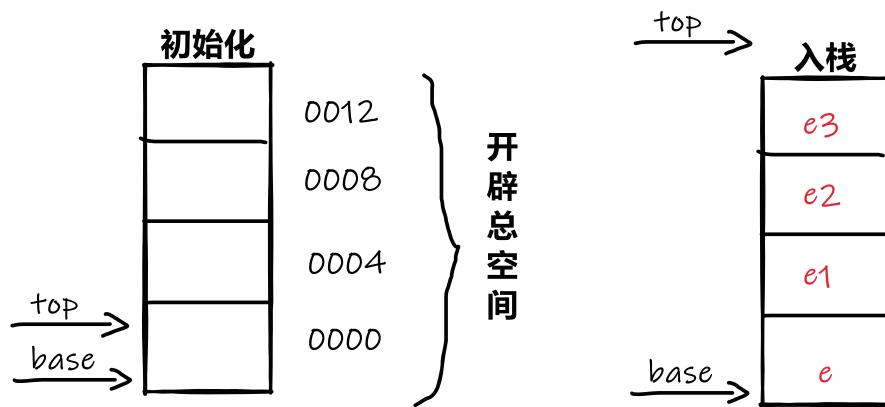
双链表



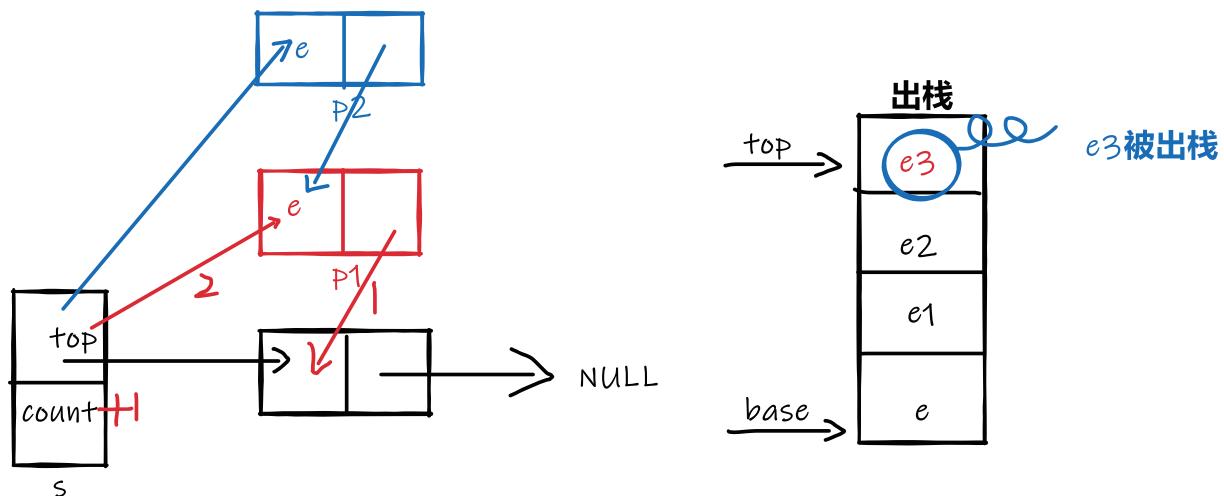
删除



顺序栈

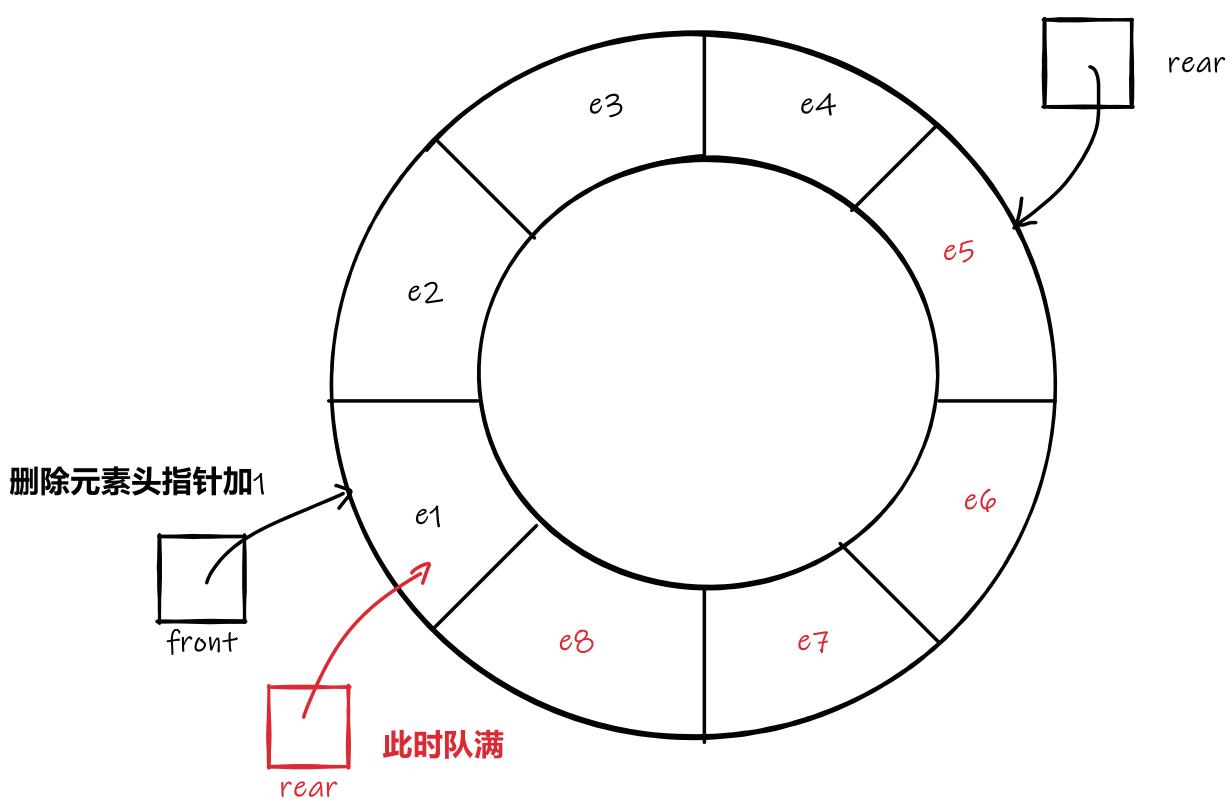


链栈



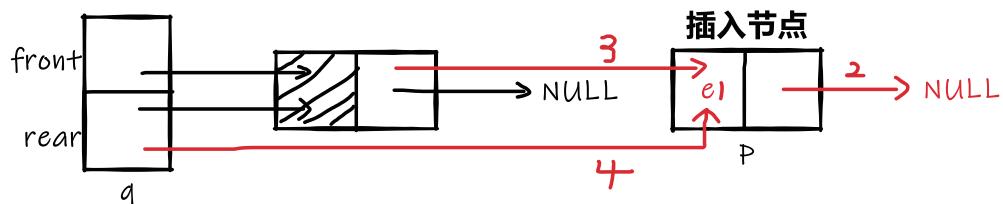
循环顺序队列

插入元素队尾指针加1

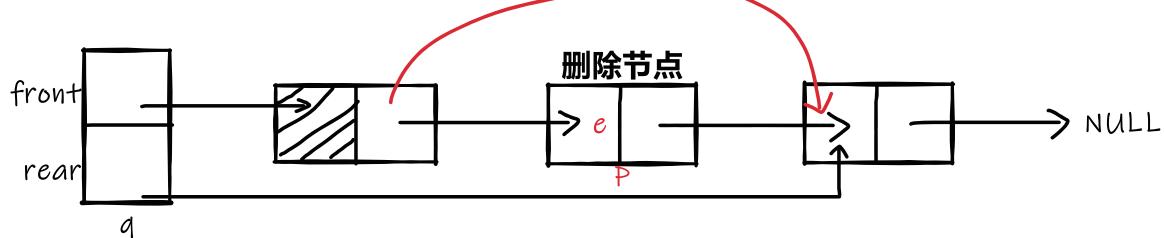


链队列

插入

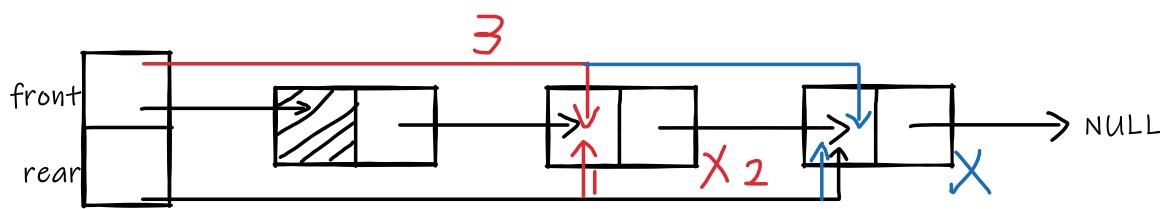


出队



如果删除节点是队尾
后一个节点，那么需
要给尾节点重新赋
值，也就是 $q->rear =$
 $q->front$

删除

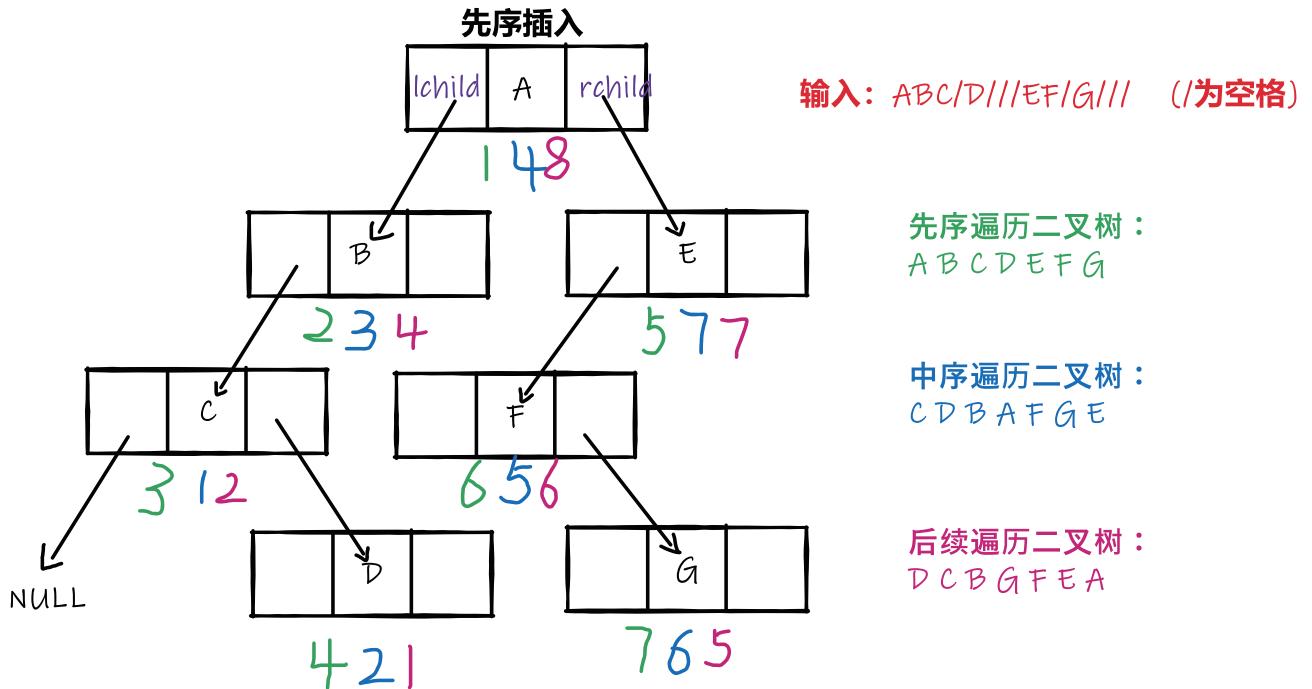


第一步

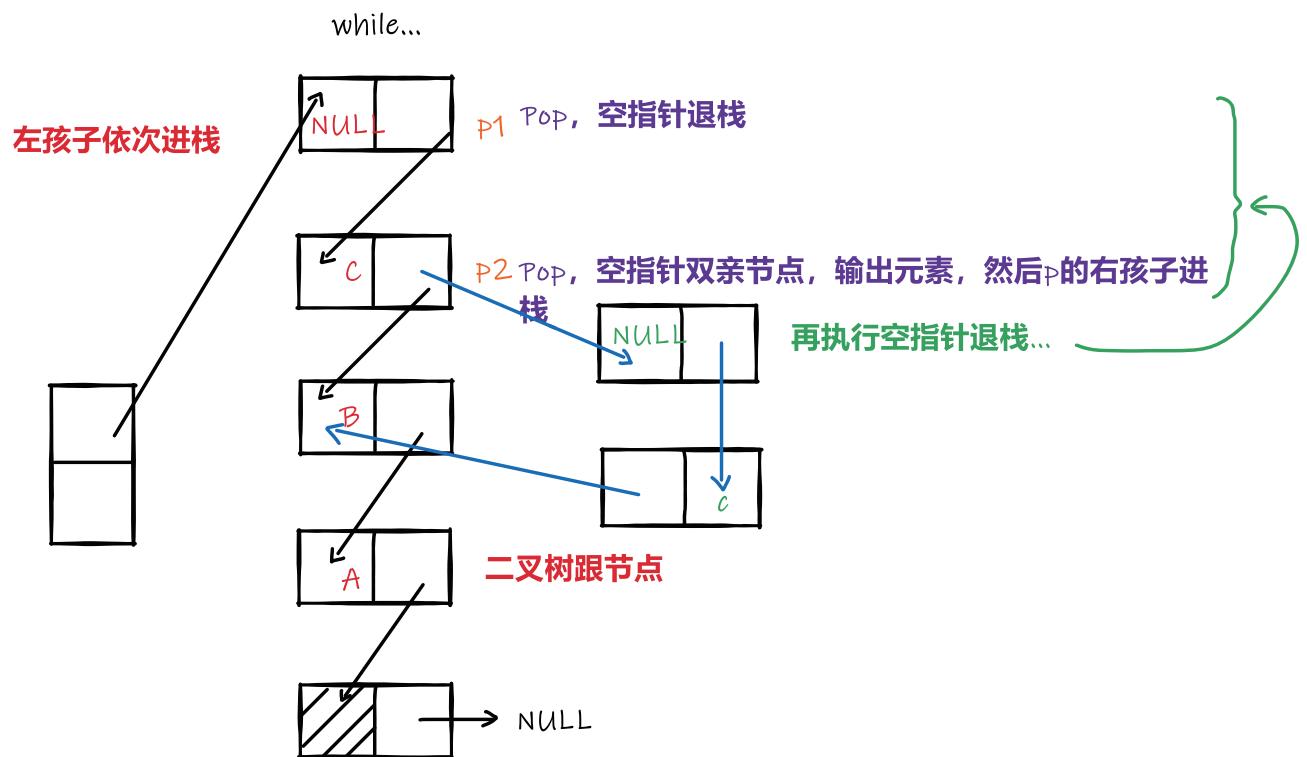
第二步

while...

树

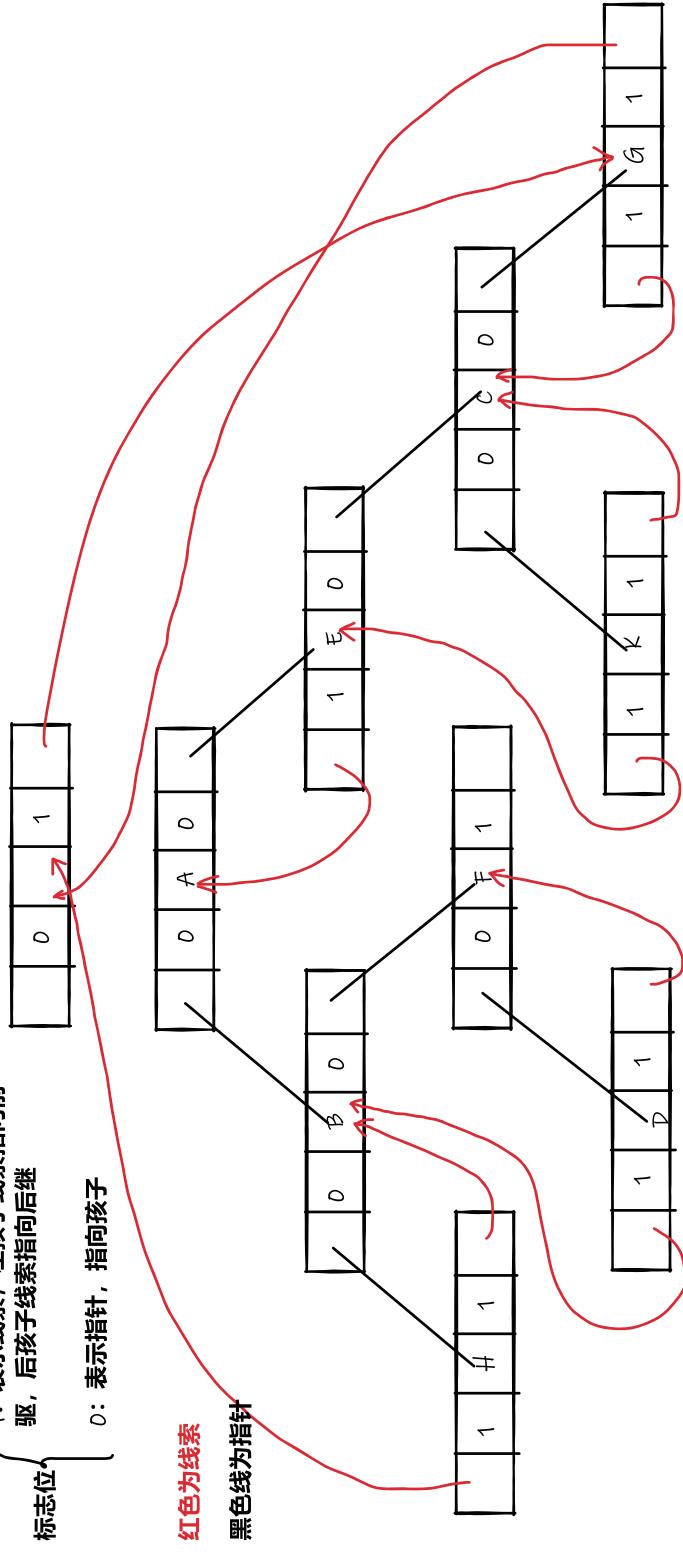


中序遍历



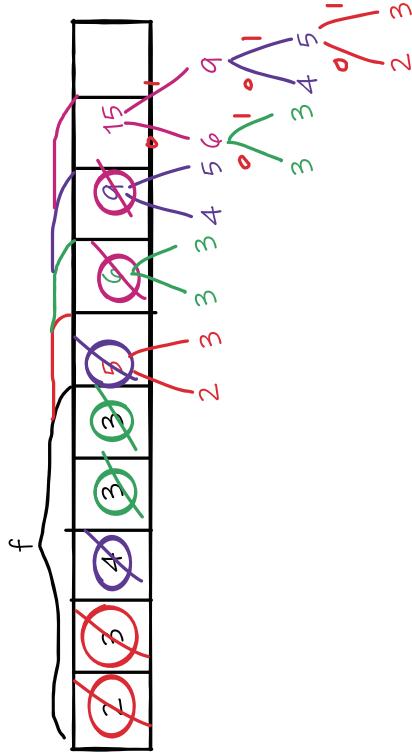
线索二叉树

标志位 $\begin{cases} 1: 表示线索, 左孩子线索指向前驱, 后孩子线索指向后继 \\ 0: 表示指针, 指向孩子 \end{cases}$



红色为线索
黑色线为指针

哈夫曼树



哈夫曼树生成:

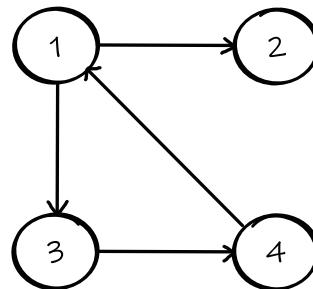
选取集合 f 中最小两位权值，组成新的二叉树，
该根节点权值为左孩子、右孩子权值之和，然
后左右在集合 f 中删除，新的二叉树放入集合 f ，
又开始寻找最小的两位权值，组成新的二叉树...
直到只剩一棵树，即为哈夫曼树

译码：

从根节点出发，扫描到的二进制串（左孩子位‘0’，右孩子位‘1’）即为叶子节点对应字符的编码，e0: ‘2’的编码：110

图

邻接矩阵



有向图

输入: 4, 4
 v1,v2=1,2
 v1,v2=1,3
 v1,v2=3,4
 v1,v2=4,1

建立邻接矩阵



	1	2	3	4
1	0	1	1	0
2	0	0	0	0
3	0	0	0	1
4	1	0	0	0

A[4][4]=

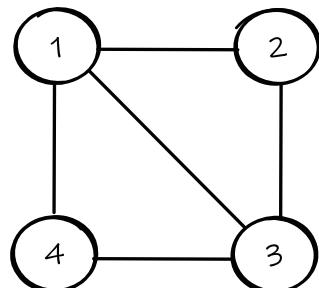
有向图邻接表

无向图邻接表以主对角线对称

指向 1 指向 2, 有方向
 <1,2>

<3,4>

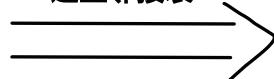
邻接表

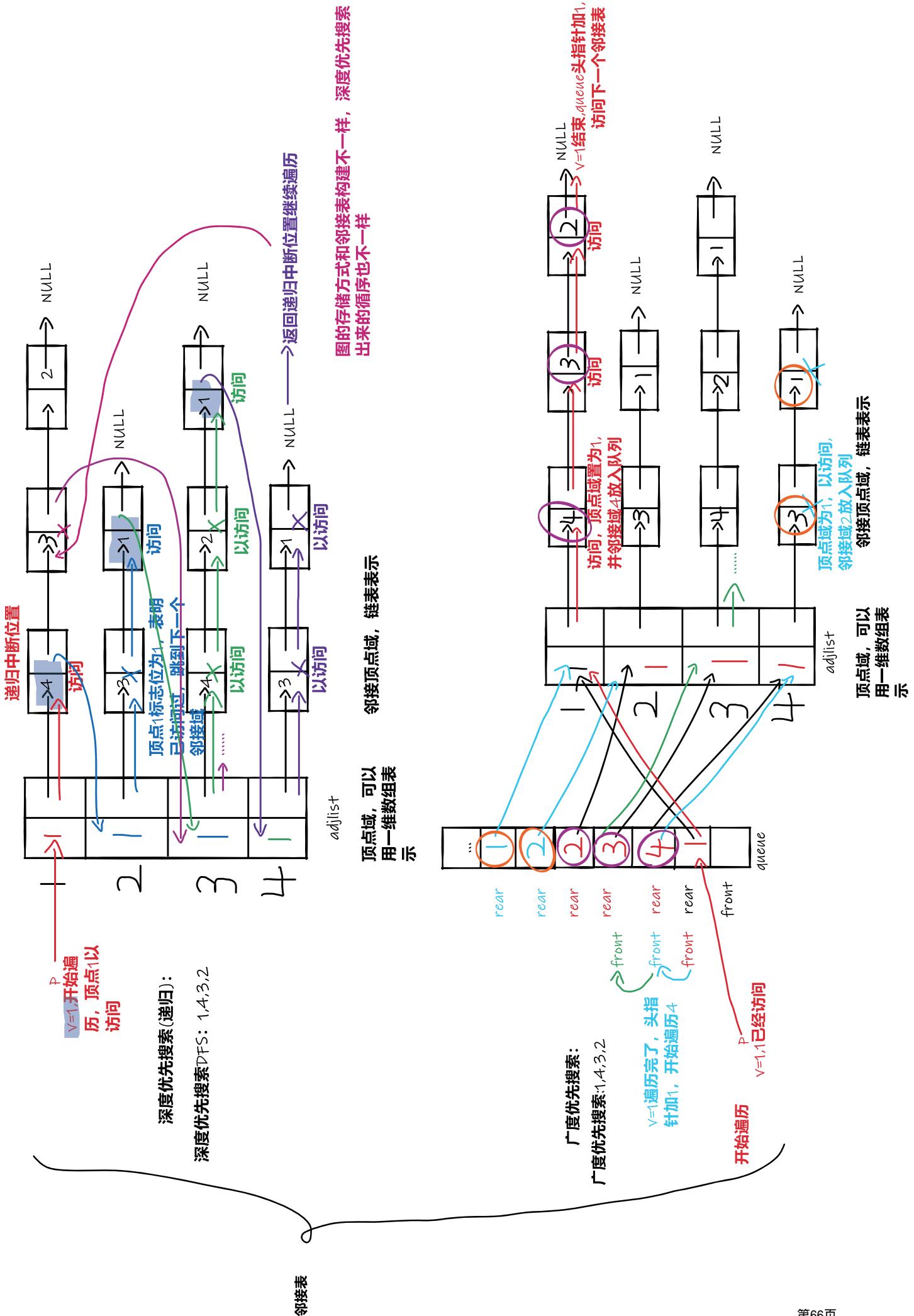


无向图

输入, 4, 5
 v1,v2=1,2
 v1,v2=1,3
 v1,v2=1,4
 v1,v2=2,3
 v1,v2=3,4

建立邻接表





邻接矩阵怎么看:

v1 通向 v2

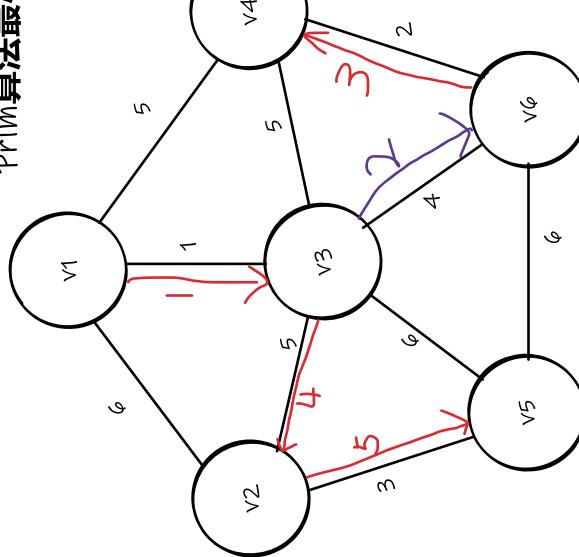
输入: (6,10)
v1,v2,w=1,2,6
v1,v2,w=1,3,1
v1,v2,w=1,4,5
v1,v2,w=2,3,5
v1,v2,w=2,5,3
v1,v2,w=3,4,5
v1,v2,w=3,5,6
v1,v2,w=3,6,4
v1,v2,w=4,6,2
v1,v2,w=5,6,6

	v1	v2	v3	v4	v5	v6
v1	32767	6	1	5	32767	32767
v2	6	32767	5	32767	3	32767
v3	1	5	32767	5	6	4
v4	5	32767	5	32767	32767	2
v5	32767	3	6	32767	32767	6
v6	32767	32767	4	2	6	32767

邻接矩阵

cost

Prim算法最小生成树



网(连通图有权值)

保存U中的各顶点与U的各顶点构成的最小权值边的权值

closevert: lowcost;

while...

第一步: 找到最小U- v_i 集合里权值最小的顶点为v1, 指向顶点3, 输出v1->3, w=1, 然后把顶点3放入U中, 再把顶点3保存顶点3

第二步: 又找U- v_i 中最小的权值4, 顶点3指向4, 指向顶点4, 输出v3->4, w=4, 然后把顶点4放入U中, 再把顶点4连接所有顶点

Kruskal克鲁斯卡尔

输入无向网的边数:10

$bV, eV, w=1, 2, 6$
 $bV, eV, w=1, 3, 5$
 $bV, eV, w=1, 6, 1$
 $bV, eV, w=2, 4, 3$
 $bV, eV, w=2, 6, 5$
 $bV, eV, w=3, 5, 2$
 $bV, eV, w=3, 6, 5$
 $bV, eV, w=4, 5, 6$
 $bV, eV, w=4, 6, 6$
 $bV, eV, w=5, 6, 4$

	0	1	2	3	4	5	6	7	8	9	10
bV	1	1	1	2	2	3	3	4	4	4	5
w	6	5	1	3	5	2	5	6	6	6	4
eV	2	3	4	6	5	6	5	6	5	6	6



克鲁斯卡尔算法
按权值从小到大排列:

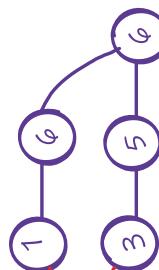
	0	1	2	3	4	5	6	7	8	9	10
bV	1	3	2	5	1	3	2	4	4	4	1
w	6	5	4	5	5	5	6	6	6	6	6
eV	2	4	6	3	6	4	5	6	5	6	2

Kruskal算法构造的最小生成树

$(1,6) 1$
 $(3,5) 2$
 $(2,4) 3$
 $(5,6) 4$
 $(2,6) 5$

记录连通分量，如果产生回路
就舍弃

初值为0，表示每个顶点自成一个分量



顶点5自成一个分量，顶点3和5组成连通分量

$set[4]=0$,
 说明顶点4
 自成一个分
 量，此时顶
 点1和6组成
 连通分量，
 $set[1]=6$

那么，1到3形成回路，舍弃

Dijkstra 迪杰斯特拉算法

设源点 $v_1 = 1$, 寻找 v_1 到其它顶点的最短路径

输入:
请输入顶点数和边数(输入格式为:顶点数,边数):6,11
 $v_1, v_2, w=1, 2, 50$
 $v_1, v_2, w=1, 3, 10$
 $v_1, v_2, w=1, 5, 45$
 $v_1, v_2, w=2, 3, 15$
 $v_1, v_2, w=2, 5, 10$
 $v_1, v_2, w=3, 1, 20$
 $v_1, v_2, w=3, 4, 15$
 $v_1, v_2, w=4, 2, 20$
 $v_1, v_2, w=4, 5, 35$
 $v_1, v_2, w=5, 4, 30$
 $v_1, v_2, w=6, 4, 3$

	0	1	2	3	4	5	6
dist	0	∞	50	10	25	45	∞
s	0	1	0	0	0	0	0
path	1	4	1	3	1	1	1

	v_1	v_2	v_3	v_4	v_5	v_6	
v_1	9999	45	10	25	45	9999	不通
v_2	9999	9999	15	9999	10	9999	
v_3	20	9999	9999	15	9999	9999	
v_4	9999	9999	9999	20	9999	9999	
v_5	9999	9999	9999	30	9999	9999	
v_6	9999	9999	3	9999	9999	9999	

算法基本思路:(权值从小找到大)

1. 先找 v_1 到其它顶点未被确定最短距离的顶点, 即: v_3, v_3 被确定最短路径, 此时 v_1 到 v_3 的权值 (w) 为 10,
2. 又开始找 v_3 到其它顶点未被确定最短距离的顶点, 假设 v_1 , 若 v_1 到 v_3 的权值 + v_3 到 v_1 的权值 $< v_1$ 到 v_1 (原来的路径) 的权值(大于说明新的路径大于原来的路径, 即表示原来的路径已经为最短路径), 则表示原来的路径已经为最短路径, 即: $v_1 = v_4$, 那么 v_1 到 v_4 的新权值为 v_1 到 v_4 的最短路径, 保存新的权值和新的前驱顶点 v_4 , 即: $dist[4] = dist[3] + cost[4]$, $path[4] = 3$

3. 重复 1,2 步, 直到查找完毕

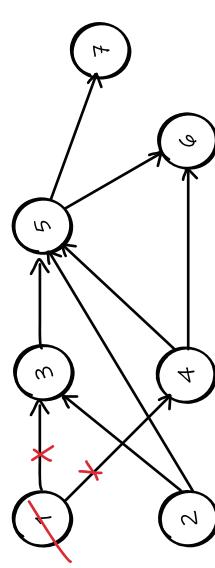
输出最短路径: path
 $2 \leftarrow 4 \leftarrow 3 \leftarrow 1 \ 45$
 $3 \leftarrow 1 \ 10$
 $4 \leftarrow 3 \leftarrow 1 \ 25$
 $5 \leftarrow 1 \ 45$
 $6 \leftarrow 1 \ \infty$

拓扑排序

检查AOV网中是否存在环

基本思想：

- 1.选着一个入度为0的顶点并且输出它
- 2.删除该项顶点并删除该点发出的全部边
- 3.重复上两个步骤,直到网中不存在没有前驱的顶点位置



输出7,6,5,4,3,2,1

v1,v2=1,3

v1,v2=1,4

v1,v2=2,3

v1,v2=2,5

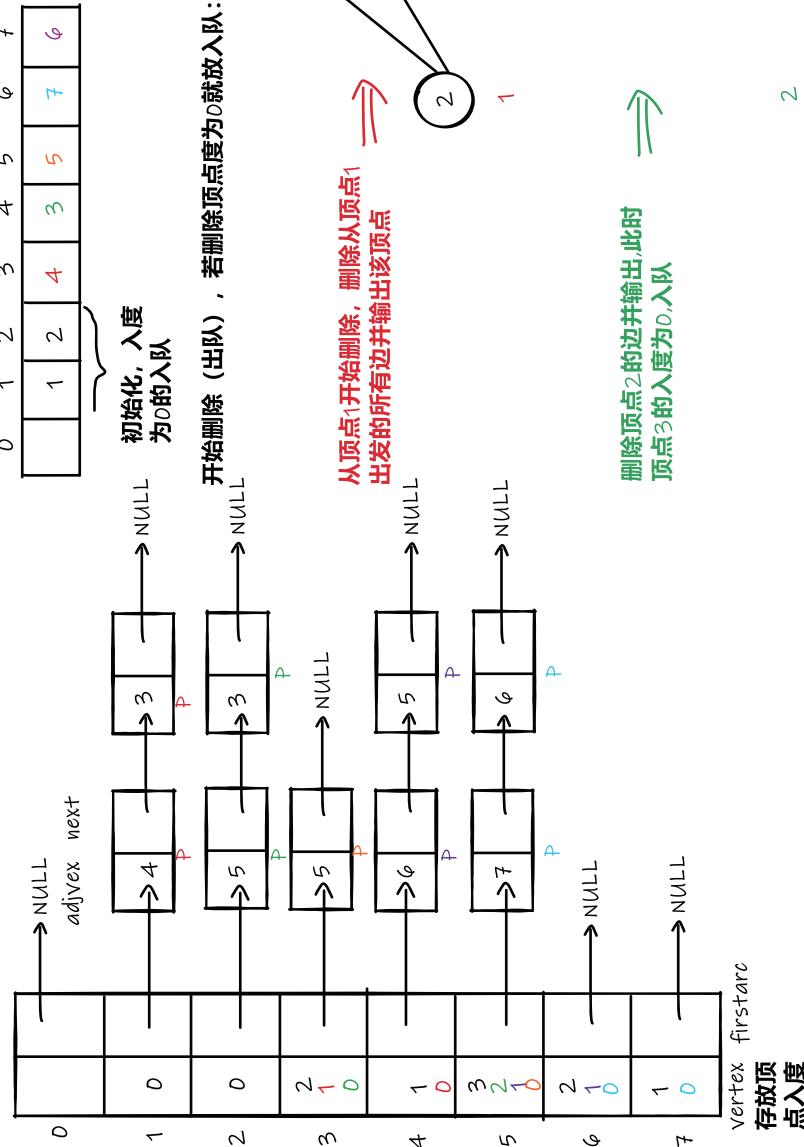
v1,v2=3,5

v1,v2=4,5

v1,v2=4,6

v1,v2=5,6

v1,v2=5,7

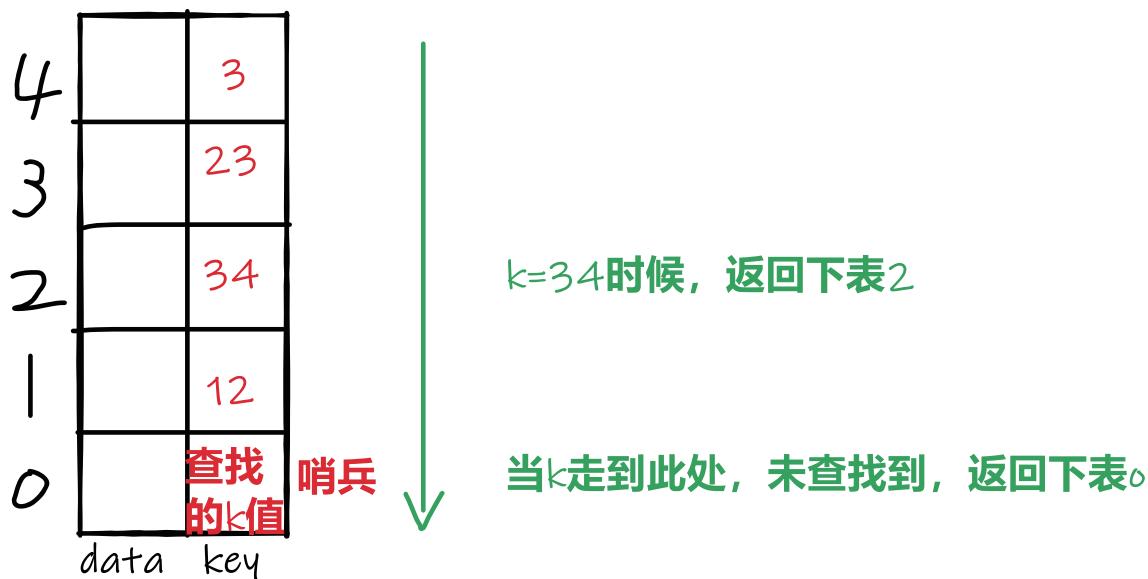


拓扑排序的结果:1243576

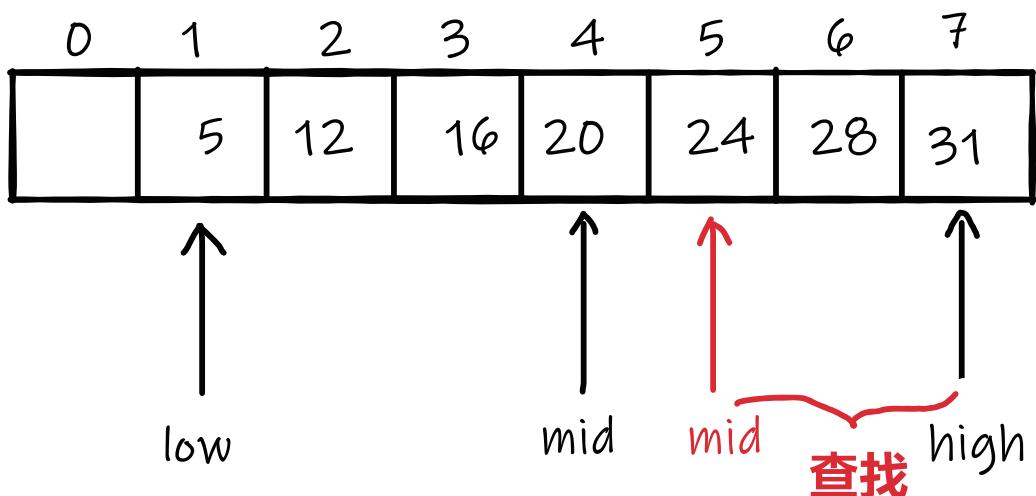
2

删除顶点1.....

顺序查找

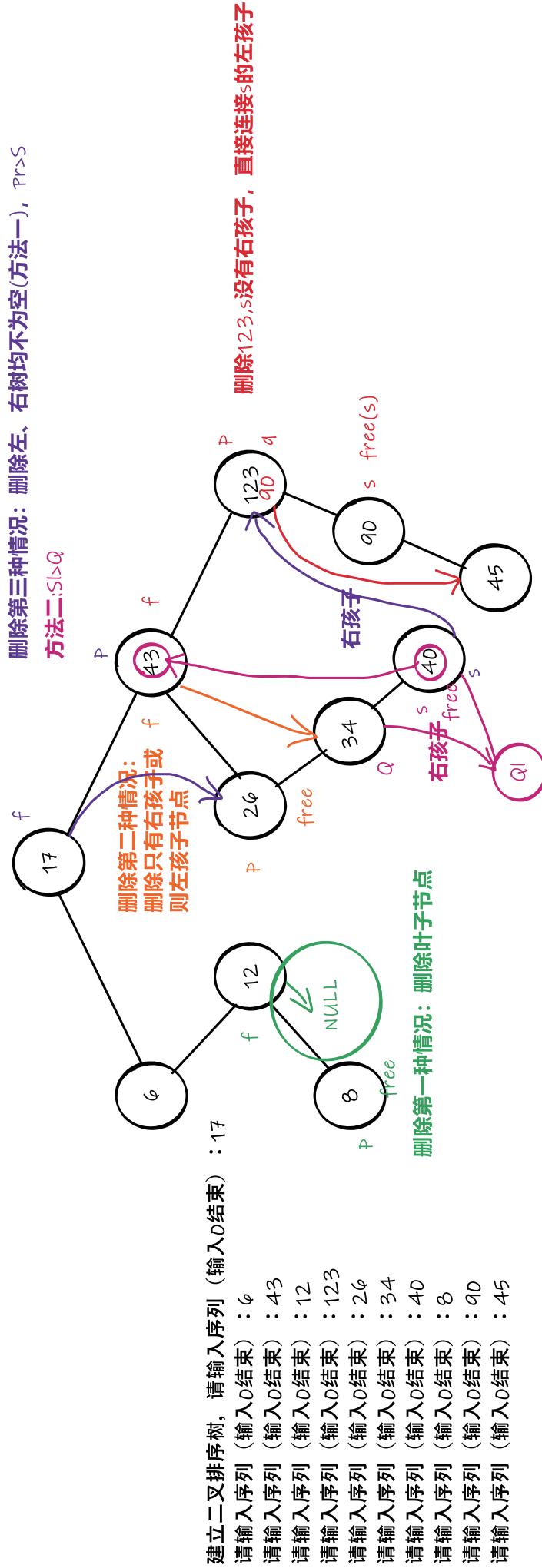


二分查找: 前提是有序表, 且顺序存储结构



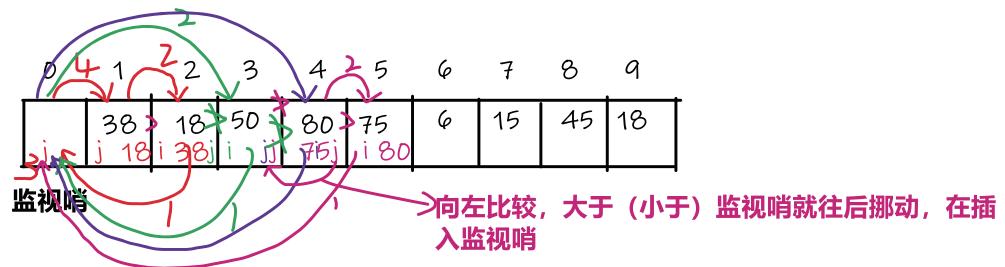
查找 $k = 24$ 时, $mid.key < k$, 那么 k 在 mid 的右边, low 重新赋值 $mid+1$, 然后继续右边查找, 此时 $mid.key$ 不小于 k , 同时也不大于 k , 说明查找到了, 返回 mid , 如果 low 大于 $high$ 说明查找表没有 k 这条记录

二叉排序树：（电子工业数据结构篇P255）

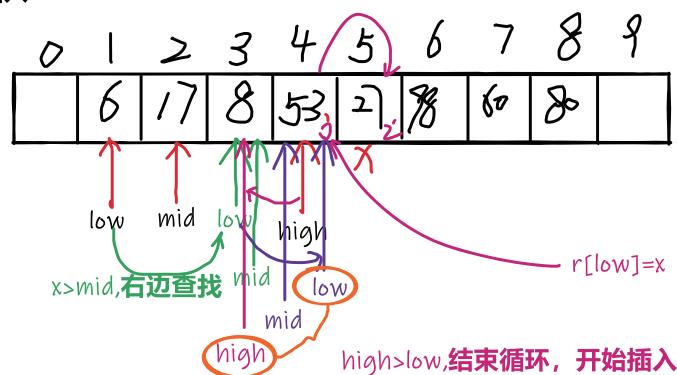


二叉排序树中序遍历：6 8 12 17 26 34 40 43 45 90 123
 删除123后的中序序列：6 8 12 17 26 34 40 43 45 90

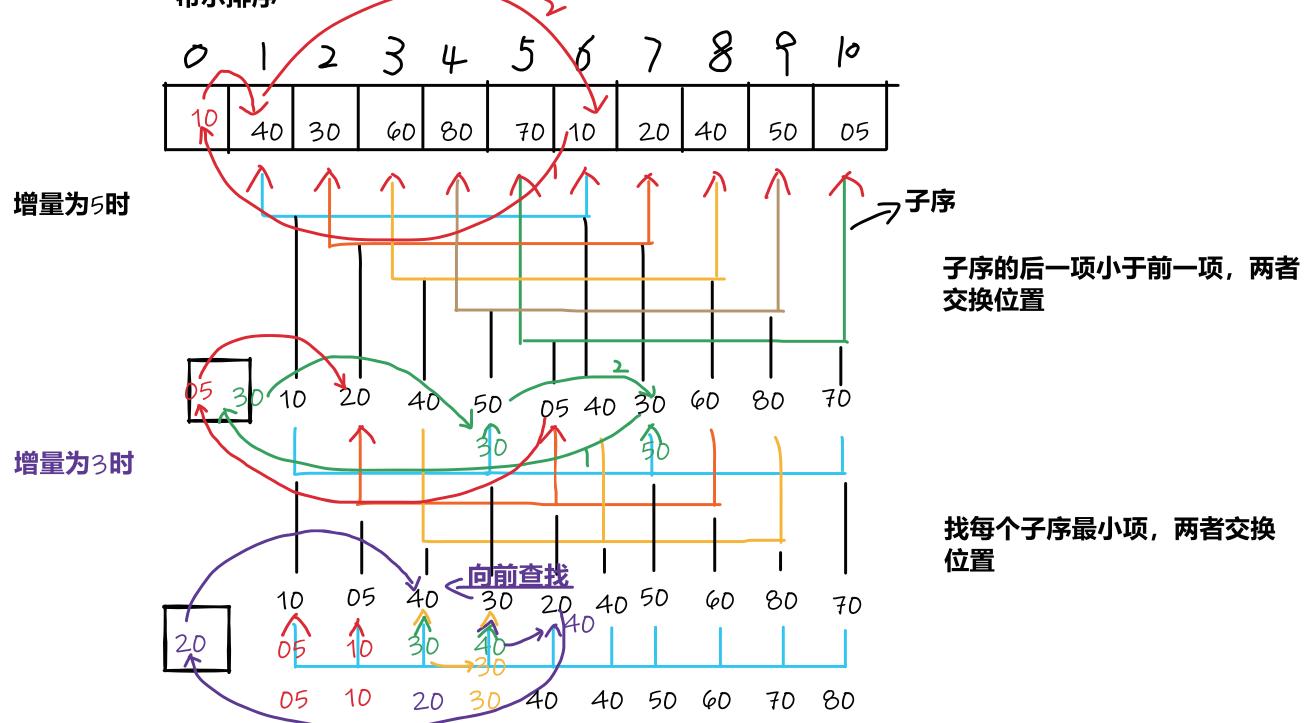
直接插入



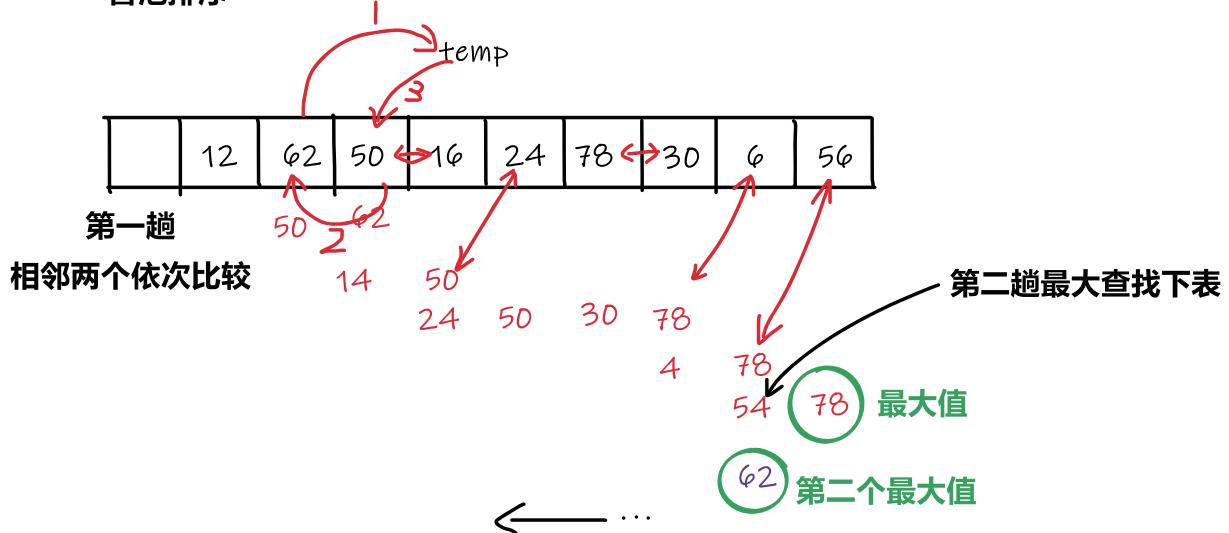
二分插入



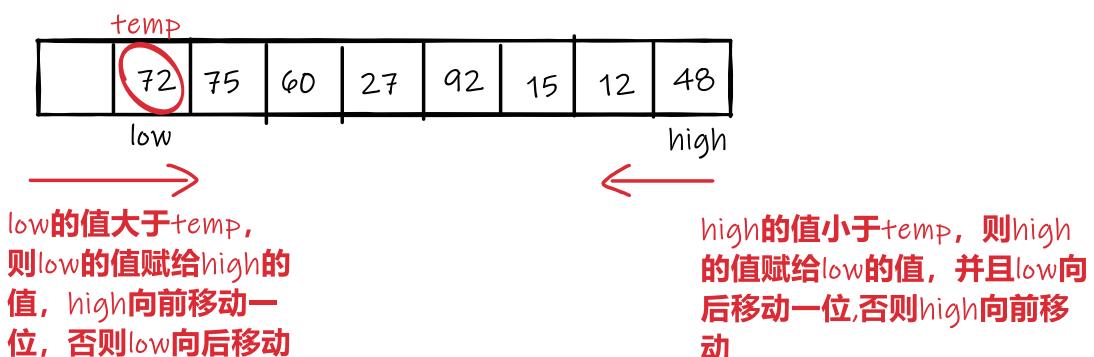
希尔排序



冒泡排序

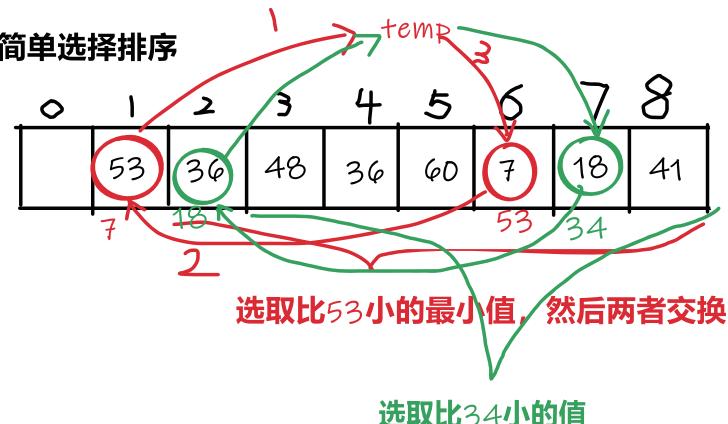


快速排序



当 $low == high$ 时, $temp$ 放回 low 的位置, 一趟完成, 返回枢轴 pos 位置, 在递归对左步子表快速排序, 再对右步子表快速排序

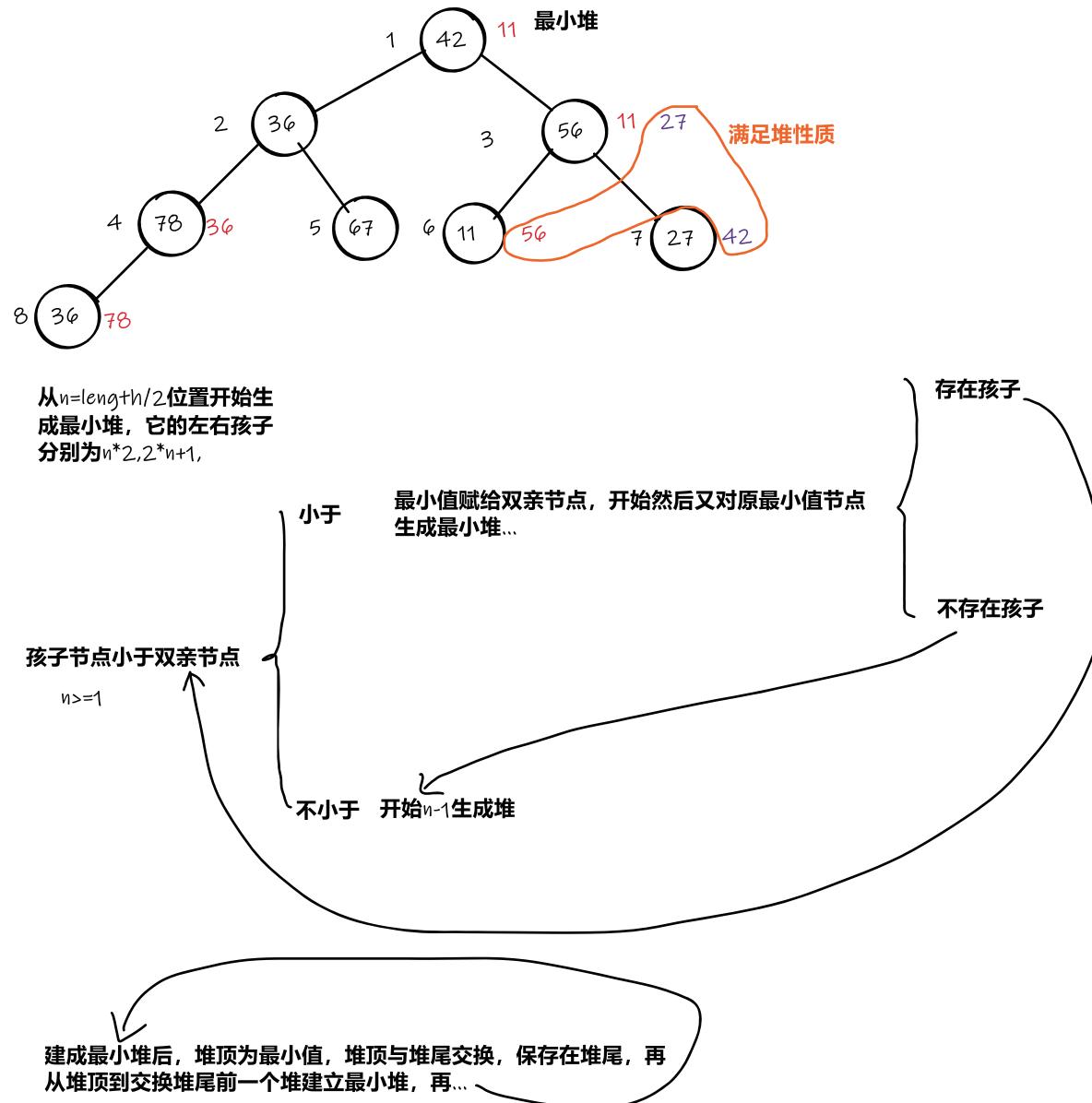
简单选择排序



选取比 53 小的最小值, 然后两者交换

选取比 34 小的值

堆排序



归并排序

