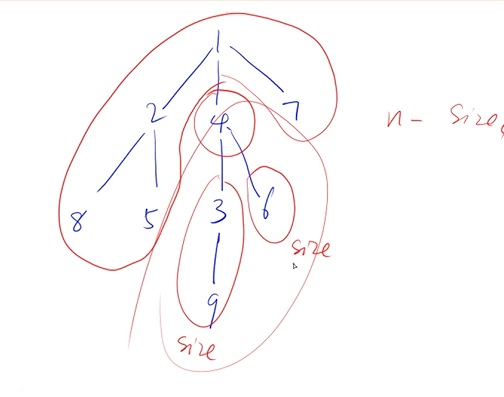
邻接表：其实是一堆链表的集合

树是一种特殊的图 所以都可以使用邻接表来存储



邻接表 用来记录没一个节点所直接连接的节点有哪些

比如：

1 ：2 4 7

2：1 5 8

3：4 9

4：1 3

5：2

6：4

7：1

8：2

理解邻接表 要先学会理解单链表 理解单链表里面每一个变量的含义

区分下标 与 编号 例如 a[1]=9;下标为1 编号为9

邻接表我们会用到

h[N] e[N] ne[N] idx

h[i]表示编号为i的节点所对应的下标 相当于头节点

e[idx]表示下标为idx的节点所对应的编号（即 节点值）

ne[idx]表示下标为idx的节点所连接的下一个节点的下标

在图与树的存储中 我们经常用到下列代码来存储

// 对于每个点k，开一个单链表，存储k所有可以走到的点。h[k]存储这个单链表的头结点

int h[N], e[N], ne[N], idx;

// 添加一条边a->b

建议用单链表的思想来理解

void add(int a, int b)

{

e[idx] = b, ne[idx] = h[a], h[a] = idx ; idx++ ;

}

// 初始化

idx = 0;

memset(h, -1, sizeof h); //必须将h数组的值全部变为 -1

遍历操作：

for(int i=1;i<=m;i++) //m个节点

{

cout<<i<<":";

for(int j=h[i];j!=-1;j=ne[j])

{

cout<<"->"<<**e[j]**;

}

cout<<endl;

}