实验舱 青少年编程 走近科学 走进名校

蛟龙五班

树

Mas

树

树是由$n(n \geq 0)$个节点组成一个**具有层次关系的**集合

把它叫做树是因为它看起来像一棵倒挂的树,也就是说它是根朝上叶朝下

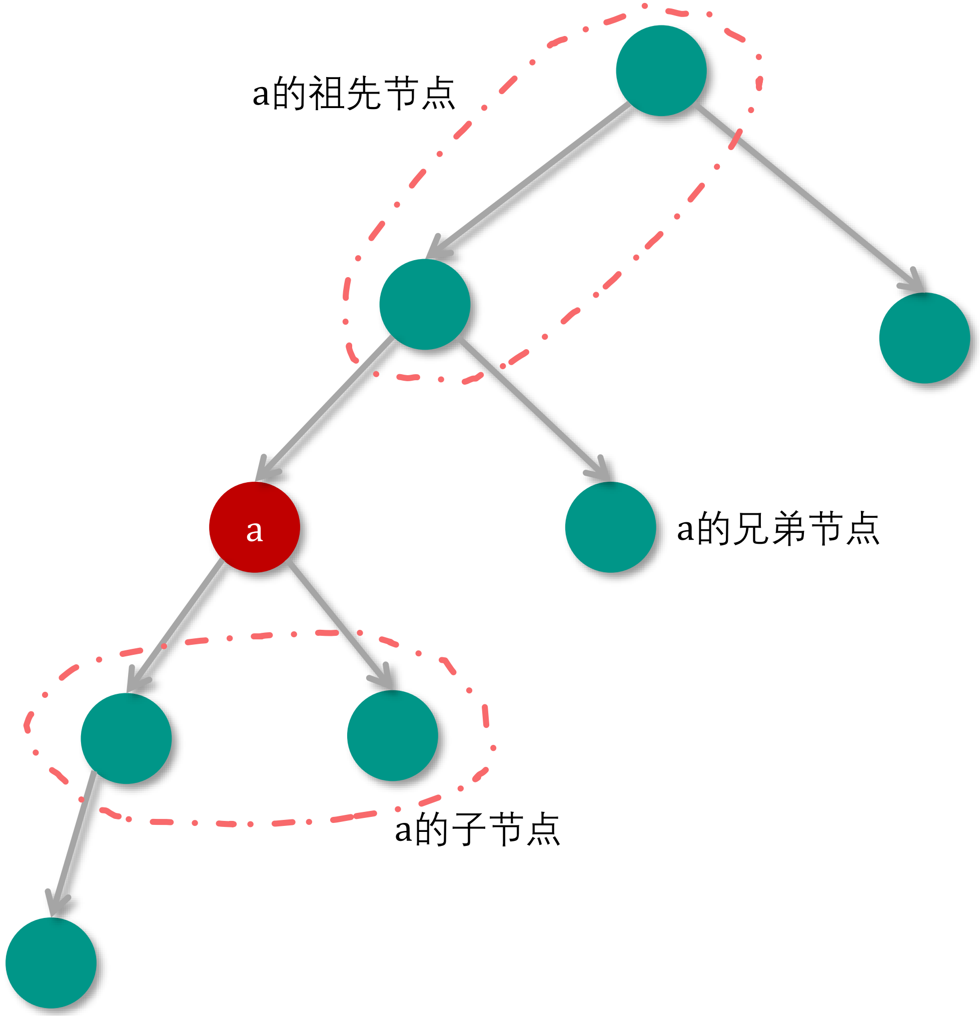
树分为有根树和无根树两种。我们通常说的树指的是有根树。

有根树具有以下的特点

- 每个节点有零个或多个子节点
- 没有父节点的节点称为根节点(就是顶上的那个唯一的点)
- 每一个非根节点只有一个父节点
- 除了根节点外,每个子节点可以分为多个不相交的子树

无根树和有根树的样子类似,但是没有“根”、“父节点”和“子节点”等概念

(大家可以想象把右图的树里每个节点都打乱)



树的性质

节点的层次($Level$)从根节点开始定义,其他节点的层次为其父节点+1

有根树的最大层次数称作整棵树的深度(Depth),树中每个节点的深度定义为以该节点为根的子树的深度

在有根树里,节点拥有的分叉数量称为节点的度(Degree)

度为 0 的节点被称为叶子节点

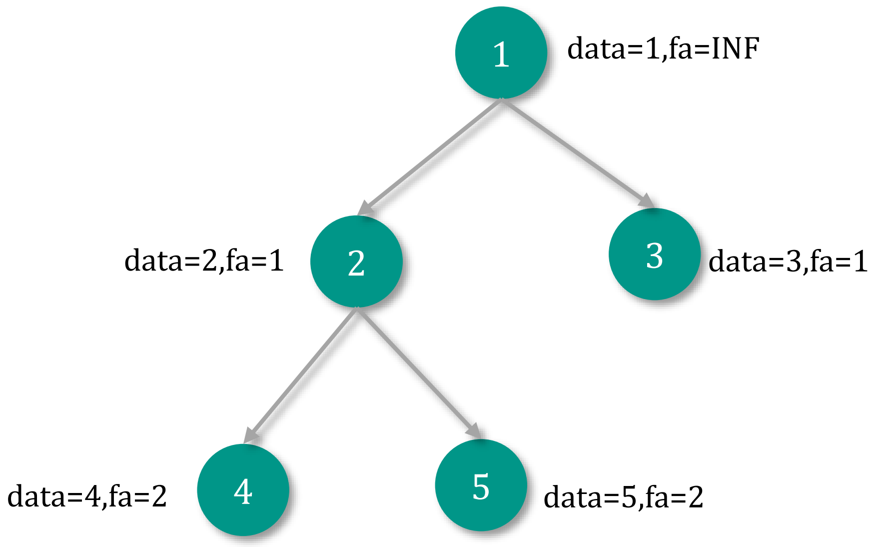
树中的节点数量 = 树的边数+1 = 所有节点度 + 1

节点度不超过为m的树,称作m叉树。

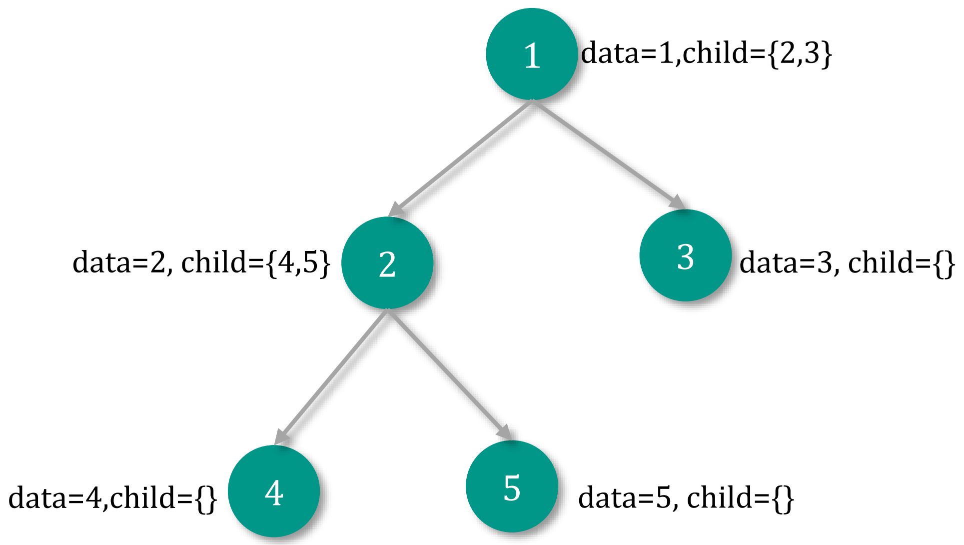
如果想通过连边的方式让 n 个节点互相连通,树是所需边数最少的方式($n-1$ 条边)

若干棵互不相交的树称作森林

树的存储



双亲表示法



孩子表示法

#1234、找树根和孩子

题目描述

给定一棵树，输出树的根 $root$，孩子最多的结点 max 以及他的孩子

输入格式

第一行：n（结点个数 ≤ 100），m (边数 ≤ 200)
以下 m 行：每行两个结点 x 和 y，表示 y 是 x 的孩子($x, y \leq 1000$)

输出格式

第一行：树根：$root$
第二行：孩子最多的结点 max
第三行：max 的孩子,按编号由小到输出

输入样例

```
8 7
4 1
4 2
1 3
1 5
2 6
2 7
2 8
```

输出样例

```
4
2
6 7 8
```

#2085 数叶子结点

题目描述

对于一棵树，可以分为叶子节点和非叶子节点，请你统计叶子节点的数量。

输入格式

第一行包含一个整数 N 表示树中结点总数以及一个整数 M 表示非叶子结点数。

接下来 M 行，每行的格式为：

```
ID K ID[1] ID[2] ... ID[K]
```

ID 是一个两位数字，表示一个非叶子结点编号，K 是一个整数，表示它的子结点数，接下来的 K 个 $ID[i]$ 也是两位数字，表示一个子结点的编号。

为了简单起见，我们将根结点固定设为 01 。

所有结点的编号即为 $01, 02, 03, \ldots, 31, 32, 33, \ldots, N$ 。

输出格式

输出从根结点开始，自上到下，树的每一层级分别包含多少个叶子节点。

输出占一行，整数之间用空格隔开。

#2025、病毒溯源

题目描述

病毒容易发生变异。某种病毒可以通过突变产生若干变异的毒株，而这些变异的病毒又可能被诱发突变产生第二代变异，如此继续不断变化。

现给定一些病毒之间的变异关系，要求你找出其中最长的一条变异链。

输入格式：

输入在第一行中给出一个正整数 N（$\leq 10^4$），即病毒种类的总数。于是我们将所有病毒从 0 到 $N-1$ 进行编号。

随后 N 行，每行按以下格式描述一种病毒的变异情况：

```
k 变异株1 …… 变异株k
```

其中 k 是该病毒产生的变异毒株的种类数，后面跟着每种变异株的编号。第 i 行对应编号为 i 的病毒（$0 \leq i < N$）。

输出格式：

首先输出从源头开始最长变异链的长度。

在第二行中输出从源头开始最长的一条变异链，编号间以 1 个空格分隔，行首尾不得有多余空格。如果最长链不唯一，则输出最小序列。

注：我们称序列 $a_1, \ldots, a_n$ 比序列 $b_1, \ldots, b_n$ 小，如果存在 $1 \leq k \leq n$ 满足 $a_i = b_i$ 对所有 $i \leq k$ 成立，且 $a_k < b_k$。

输入样例：

```
10
3 6 4 8
0
0
0
2 5 9
0
1 7
1 2
0
2 3 1
```

输出样例：

```
4
0 4 9 1
```

#2025、病毒溯源

可采用孩子表示法存储树

树上可能存在多条传播链,传播链的源头入度为0

dfs 找出最长路径,并记录最长路径

对于类字典序输出最小的方案,仅需要保证遍历时从节点编号从小到大开始

```cpp
void dfs(int u, int deep)
{
  if (maxLen < deep)
  {
    maxLen = deep;
    if (!tree[u].size())
      idx = u;
  }
  d[u] = deep;
  sort(tree[u].begin(), tree[u].end());
  for (auto &&i : tree[u])
    dfs(i, deep + 1);
}
```

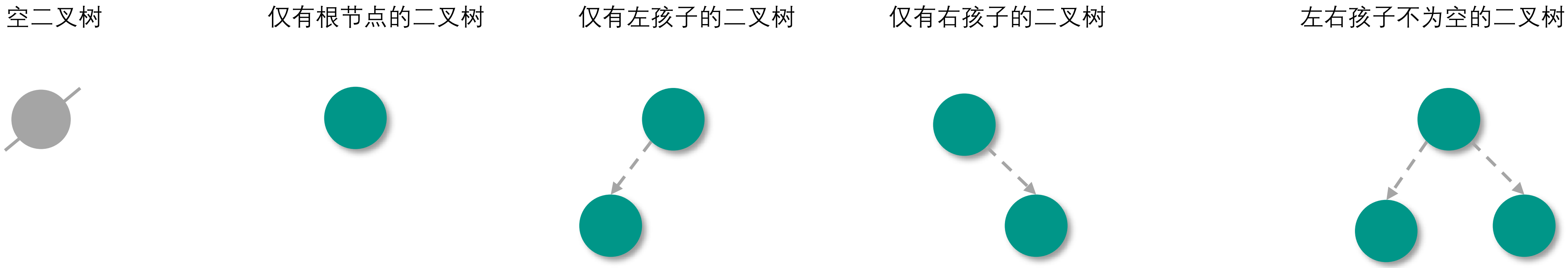
二叉树

二叉树(binary tree)是一种特殊的树型结构,树中节点的度最大为2

即二叉树的每个结点最多有两个子结点

每个结点的子结点分别称为左孩子、右孩子,它的两棵子树分别称为左子树、右子树

二叉树有 5 种基本形态:



#1866、二叉树的深度和宽度

题目描述

有一颗二叉树，如下：

```
       1
     /  \
    2    3
   / \   /
  4   5  6
          \
           7
```

此二叉树共有 7 个结点 $1 \sim 7$ ，并约定结点 1 为根结点，处在第一层。根结点 1 有 2 个孩子，左孩子为 2 ，右孩子为 3 ，并约定二叉树的一个结点最多有 2 个孩子。
二叉树可以用三元式表示，（ `结点 左孩子 右孩子` ）

输入

第一行一个整数 n ，即结点个数。

接下来 n 行，每行三个数，即结点三元式。

输出

一行，含二个整数，即二叉树深度和宽度

数据规模

对于全部的数据 $1 \leq n \leq 10^5$

输入#1

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

输出#1

```
3 2
```

#1866、二叉树的深度和宽度

树的存储采用孩子表示法存储,仅需要存储左右孩子编号

宽度即所有层次中的节点数量最大值

高度即为最大层次数+1

可 dfs/bfs 求出

```cpp
void bfs()
{
    queue<int> q;
    q.push(1);
    while (q.size())
    {
        int cnt = q.size();
        w = max(w, cnt);
        h++;
        for (int i = 0; i < cnt; i++)
        {
            int root = q.front();
            q.pop();
            if (tree[root].l)
                q.push(tree[root].l);
            if (tree[root].r)
                q.push(tree[root].r);
        }
    }
}
```

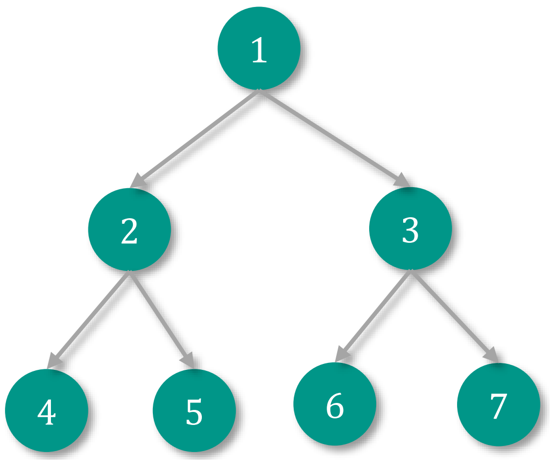
特殊二叉树

完全二叉树(complete binary tree)

只有最下面两层结点的度数可以小于2,且最下面一层的结点都集中在该层最左边的连续位置上

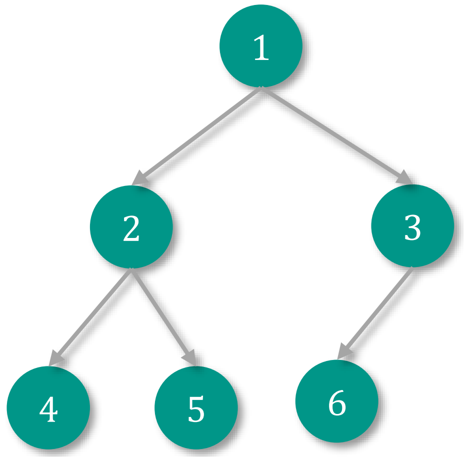
满二叉树(full/proper binary tree)

每个结点的子结点数量均为0或者2的二叉树。换言之,每个结点或者是树叶,或者左右子树均非空



满二叉树

完全二叉树就是满二叉树去掉最后一排的一段右连续的点!



完全二叉树

二叉树性质

在二叉树的第 i 层上最多有 2^{i-1} 个结点($i \geq 1$)

高度为 h 的二叉树节点数量最多为 $2^h - 1$

具有 n 个结点的完全二叉树的深度为 $\lfloor \log_2 n \rfloor + 1$

对任意一棵二叉树,如果其叶结点数为n_0,度为2的结点数为 n_2 则有

$$n_0 = n_2 + 1$$

对于一棵n个结点的完全二叉树,对任一个结点 i 有

若 $i = 1$,则结点 i 为根,无父结点; 如果$i > 1$,则其父结点编号为 $\left\lfloor \frac{i}{2} \right\rfloor$

若 $2 \times i > n$,则结点 i 无左孩子,否则左孩子编号为 $2 \times i$

若 $2 \times i + 1 > n$,则结点 i 无右孩子,否则右孩子编号为 $2 \times i + 1$

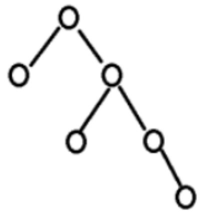
如何证明?

二叉树性质

(2018普及组初赛) 根节点深度为0,深度为h的满$k(k > 1)$叉树,即除最后一层无子节点,其他每层所有结点都有k个子结点的树,共有( )个结点

$A.\frac{k^{h+1}-1}{k-1}$ $B.k^{h-1}$ $C.k^h$ $D.\frac{k^h}{k-1}$

(2019 $CSP-J$初赛) 一棵二叉树如右图所示,若采用顺序存储结构,即用一维数组元素存储该二叉树中的结点(根结点的下标为1,若某结点的下标为i,则其左孩子位于下标$2i$处、右孩子位于下标$2i+1$处),则该数组的最大下标至少为( )



$A.6$ $B.10$ $C.15$ $D.12$

(2020 $CSP-J$初赛) 独根树的高度为 1,具有61个结点的完全二叉树的高度为?

$A.7$ $B.8$ $C.5$ $D.6$

(2021 $CSP-J$初赛) 一棵二叉树只有根结点,那么这棵二叉树高度为1。高度为5的完全二叉树有( )种不同的形态?

$A.16$ $B.15$ $C.17$ $D.32$

(2013普及组初赛) 已知一棵二叉树有10个节点,其中至多有( )个节点有2个节点

$A.4$ $B.5$ $C.6$ $D.7$

#101、二叉树的第k层

题目描述

有一棵树，输出某一深度的所有节点，有则输出这些节点，无则输出 `EMPTY` 。该树是完全二叉树

输入格式

输入一个 $n(1 \le n \le 1000)$ ，然后将树中的这 n 个节点依次输入，再输入一个 k 代表深度

输出格式

输出该树中第 k 层得所有节点,节点间用空格隔开

样例输入1

```
5
1 2 3 4 5
7
```

样例输入2

```
7
1 2 3 4 5 6 7
2
```

样例输出1

```
EMPTY
```

样例输出2

```
2 3
```

对于完全二叉树,可直接使用数组存储

根节点下标为1

第k层的下标范围为$[2^k, \min(2^{k+1}-1, n)]$

```
cin >> n;
for (int i = 1; i <= n; i++)
  cin >> tree[i];
cin >> k;
s = 1 << --k;
if (s > n)
  cout << "EMPTY";
for (int i = s; i <= min(n, s * 2 - 1); i++)
  cout << tree[i] << " ";
```

二叉树遍历

前序遍历

先处理当前节点,再依次递归处理左子树、右子树$A\ B\ D\ E\ C\ F\ G$

中序遍历

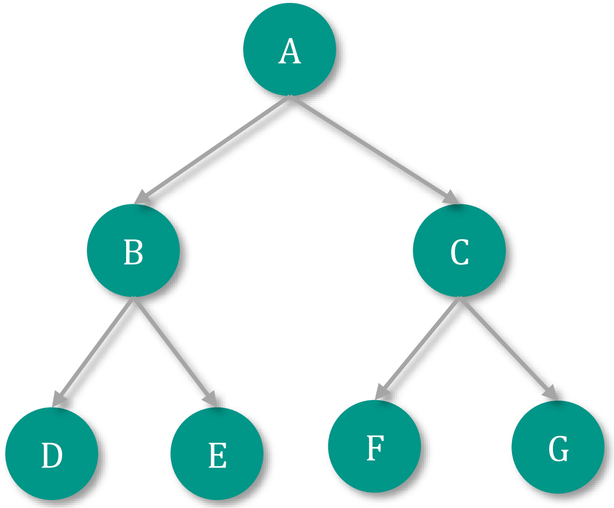
先递归处理左子树,再处理当前节点,最后递归处理右子树$D\ B\ E\ A\ F\ C\ G$

后序遍历

先递归处理左子树,再递归处理右子树,最后处理当前节点,$D\ E\ B\ F\ G\ C\ A$

层序遍历

按照层次进行遍历,$A\ B\ C\ D\ E\ F\ G$

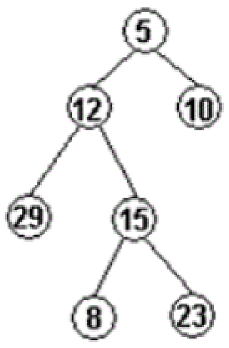


#2088、树的前中后序遍历

【题目描述】

如图二叉树的数据文件的数据格式如下:

```
7
15
5 2 3
12 4 5
10 0 0
29 0 0
15 6 7
8 0 0
23 0 0
```



【输入】

第一行 n 为二叉树的结点个树， $n \leq 100$;

接下来 n 行，每行第一列数据是各结点的值，第二列数据是左儿子结点编号，第三列数据是右儿子结点编号。

根节点编号为 1

【输出】

输出三行分别是前序遍历、中序遍历、后序遍历的结果

```cpp
void preOrder(int root)
{
    if (root > n || !root)
        return;
    cout << tree[root].data << " ";
    preOrder(tree[root].l);
    preOrder(tree[root].r);
}
void inOrder(int root)
{
    if (root > n || !root)
        return;
    inOrder(tree[root].l);
    cout << tree[root].data << " ";
    inOrder(tree[root].r);
}
void postOrder(int root)
{
    if (root > n || !root)
        return;
    postOrder(tree[root].l);
    postOrder(tree[root].r);
    cout << tree[root].data << " ";
}
```

#1233、求后序遍历

【题目描述】

输入一棵二叉树的先序和中序遍历序列，输出其后序遍历序列。

【输入】

共两行，第一行一个字符串，表示树的先序遍历，第二行一个字符串，表示树的中序遍历。树的结点一律用小写字母表示。

【输出】

一行，表示树的后序遍历序列。

【输入样例】

```
abdec
dbeac
```

【输出样例】

```
debca
```

【数据规模】

对于全部的数据字符串长度不超过 50 ,保证数据合法

#1233、求后序遍历

前序序列的第一个是根

　　中序遍历根的前方为左孩子及其子树的中序序列

　　中序遍历根的后方为右孩子及其子树的中序序列

　　对于每一个子树可以看成一个全新的树,仍然遵循上面的规律

```
int build(int l, int r)
{
  if (l > r)
    return -1;
  int root = ++idx; //给出新节点编号
  tree[root].data = pre[pos++];
  int t = in.find(tree[root].data);
  tree[root].l = build(l, t - 1);
  tree[root].r = build(t + 1, r);
  return root;
}
```

若已知中后求前?

　　后序序列的最后一个是根

　　中序遍历根的前方为左孩子及其子树的中序序列

　　中序遍历根的后方为右孩子及其子树的中序序列

　　对于每一个子树可以看成一个全新的树,仍然遵循上面的规律

若已知前后,能否唯一确定一颗二叉树的形态?

谢谢观看