合肥大学 HEFEI UNIVERSITY

数据结构与算法

人工智能与大数据学院

本节课签到码

① 计算题：一棵非空二叉树的第 3 层最多有多少个结点？

② 计算题：深度为 3 的二叉树至多有多少个结点？

③ 计算题：具有100个结点的完全二叉树的深度是多少？

④ 计算题：一棵具有129个结点的完全二叉树，结点64是否有右孩子？（填0或1）

本节课内容安排

一、二叉树的遍历算法及应用

二、树与二叉树之间的转换

三、哈夫曼树及其应用

二叉树的遍历

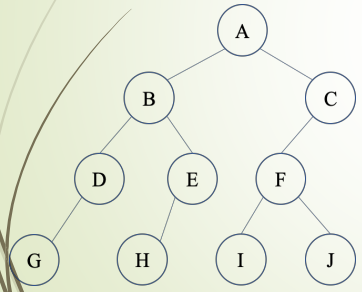
已知一棵二叉树：
先序遍历序列为{A, B, D, G, E, H, C, F, I, J}
中序遍历序列为{G, D, B, H, E, A, I, F, J, C}

构造出这棵二叉树的结构，同时给出此二叉树的后序遍历序列

二叉树的遍历

已知一棵二叉树：
先序遍历序列为{A, B, D, G, E, H, C, F, I, J}
中序遍历序列为{G, D, B, H, E, A, I, F, J, C}

构造出这棵二叉树的结构，同时给出此二叉树的后序遍历序列



G, D, H, E, B, I, J, F, C, A

二叉树的遍历

一棵二叉树的先序遍历序列、中序遍历序列、后序遍历序列如下所示，其中一部分未标出。请补全序列中的结点并构造出该二叉树。

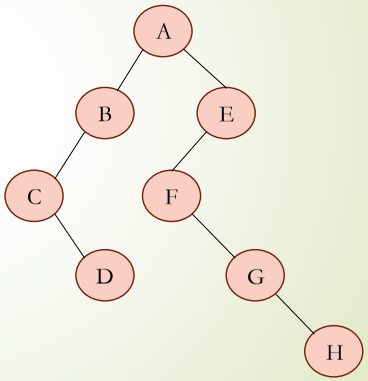
先序遍历序列：{?, ?, C, D, E, ?, G, H};
中序遍历序列：{C, D, ?, A, ?, G, H, E};
后序遍历序列：{?, C, B, ?, G, F, ?, A};

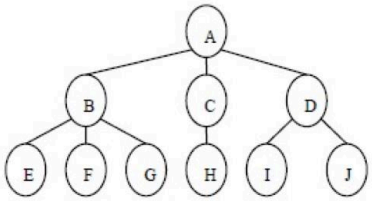
二叉树的遍历

一棵二叉树的先序遍历序列、中序遍历序列、后序遍历序列如下所示，其中一部分未标出。请补全序列中的结点并构造出该二叉树。

先序遍历序列：{?, ?, C, D, E, ?, G, H};
中序遍历序列：{C, D, ?, A, ?, G, H, E};
后序遍历序列：{?, C, B, ?, G, F, ?, A};

先序遍历序列：ABCDEFGH
中序遍历序列：CDBAFGHE
后序遍历序列：DCBHGFEA





(a) 树

树的遍历？

先序遍历：先访问树的根结点，再依次先序遍历根的每棵子树（从左到右）
序列为：{A, B, E, F, G, C, H, D, I, J}

后序遍历：先依次遍历根的每棵子树（从左到右），最后访问根结点。
序列为：{E, F, G, B, H, C, I, J, D, A}

层次遍历：按照从上到下，从左到右的顺序访问树的每一个结点。
序列为：{A, B, C, D, E, F, G, H, I, J}

树与二叉树的相互转换

树或森林与二叉树之间有一个自然的一一对应关系。任何一个森林或一棵树可惟一地对应到一棵二叉树；反之，任何一棵二叉树也能惟一地对应到一个森林或一棵树。

树与二叉树的相互转换

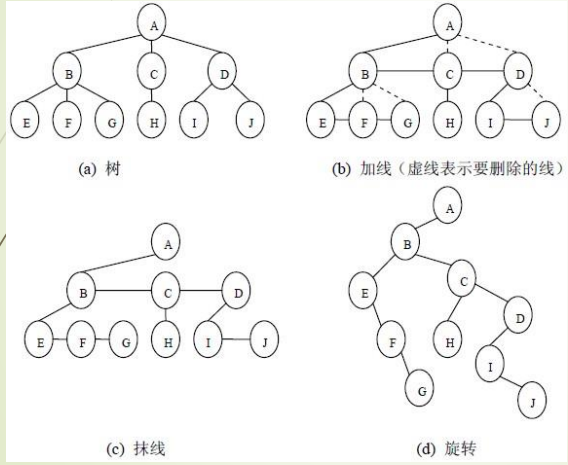
要将一棵树转换成二叉树，只需进行如下三步操作：

（1）**加线**：将兄弟结点用线相连

（2）**去线**：只保留双亲与最左边孩子结点的连线，去掉双亲和其他孩子结点的连线

（3）**旋转**：将经过加线和去线以后的结果，以树的根结点为轴心，将整棵树顺时针旋转一定角度，使之结构层次分明进行旋转处理得到转换后的二叉树

树与二叉树的相互转换



树与二叉树的相互转换

反过来，要将一棵二叉树转换成树，采用类似的三个步骤：

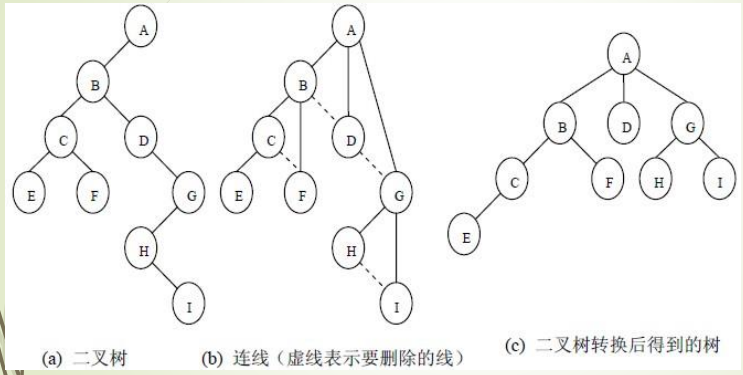
（1）**加线**：将结点和其左孩子结点的右孩子结点以及该右孩子结点的右孩子结点加线粗连

（2）**去线**：去掉结点和右孩子结点的连线

（3）**旋转**：将加线、去线后的结果进行旋转处理，就得到转换后的树

树与二叉树的相互转换

二叉树转换为树的过程如下所示：



(a) 二叉树　(b) 连线（虚线表示要删除的线）　(c) 二叉树转换后得到的树

（1）**加线**：将结点和其左孩子结点的右孩子结点以及该右孩子结点的右孩子结点加线粗连

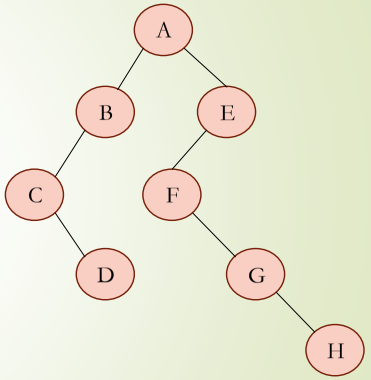
（2）**去线**：去掉结点和右孩子结点的连线

（3）**旋转**：将加线、去线后的结果进行旋转处理，就得到转换后的树

哈夫曼树

（1）基本概念

- **路径**：从树中一个结点到另一个结点之间的分支构成这两个结点之间的路径
- **路径长度**：路径上的分支数目称为路径长度
- **树的路径长度**：从树的根结点到每一个结点的路径长度之和称为树的路径长度
- **结点的带权路径长度**：从结点到根结点之间的路径长度乘以结点的权值可以得到结点的带权路径长度
- **树的带权路径长度**：树中所有叶子结点的带权路径长度之和称为树的带权路径长度，记作$WPL=\sum_{k=1}^{n} w_k l_k$（其中$w_k$为权值，$l_k$为结点到根结点的路径长度）



哈夫曼树

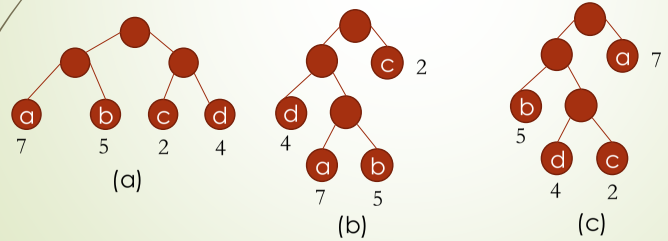
（2）最优二叉树（哈夫曼树）的定义

假设有n个权值$\{w_1, w_2, \ldots, w_n\}$，构造一棵有n个叶子结点的二叉树，使得每个叶子结点的权值为w_i，则其中带权路径长度最小的二叉树称为最优二叉树或者哈夫曼树

哈夫曼树

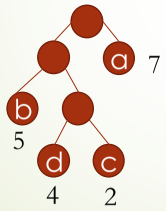
例如，下图中的3棵二叉树都有4个叶子结点a,b,c,d。它们的权重分别为7,5,2,4,三棵树的带权路径长度分别为

$(a)WPL=7\times2+5\times2+2\times2+4\times2=36$ $(b)WPL=7\times3+5\times3+2\times1+4\times2=46$
$(c)WPL=7\times1+5\times2+2\times3+4\times3=35$



上图所示的三棵二叉树，其中图（c）的带权路径长度最小。可以验证，它就是一棵最优二叉树（哈夫曼树）。也就是说，所有以权重分别为7，5，2，4的结点作为叶子的二叉树中。最小带权路径长度为35。

$(c)WPL = 7\times1 + 5\times2 + 2\times3 + 4\times3 = 35$



(c)

哈夫曼树算法

怎样根据n个权值$\{w_1, w_2, \ldots, w_n\}$，构造哈夫曼树呢？
哈夫曼算法可以很好解决这个问题。算法思想为：

（1）根据给定的n个权值$w_1, w_2, \ldots, w_n$，构成n棵二叉树的集合F=$\{T_1, T_2, \ldots, T_n\}$，其中每棵二叉树$T_i$中只有一个权值为$w_i$的根结点，其左右子树均为空。

（2）在F中任选两棵根结点的权值最小的树作为左右子树，构成一棵新的二叉树且置新的二叉树的根结点的权值为其左右子树上根结点的权值之和。

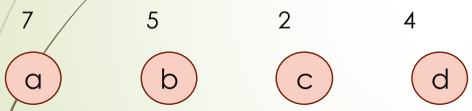
（3）从F中删除这两棵树，同时将新得到的二叉树加入到F中。

（4）重复（2）和（3）步，直到F中只含一棵树为止。这时，这棵树便是最优二叉树，即哈夫曼树

哈夫曼树算法——哈夫曼树的构造

例题1：有4个结点a, b, c, d，权值分别为7, 5, 2, 4，请构造一棵哈夫曼树。

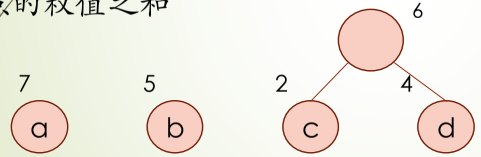
（1）构造二叉树的集合F = $\{T_1, T_2, T_3, T_4\}$



哈夫曼树算法——哈夫曼树的构造

例题1：有4个结点a, b, c, d，权值分别为7, 5, 2, 4，请构造一棵哈夫曼树。

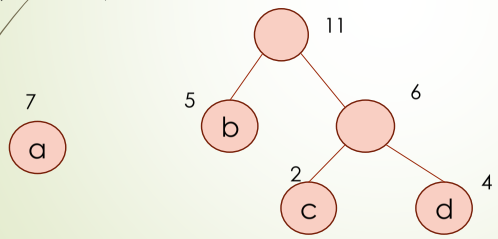
（2）在F中任意选两棵根结点的权值最小的树作为左右子树，构成一棵新二叉树，且新二叉树的根结点的权值等于左右子树上根结点的权值之和



哈夫曼树算法——哈夫曼树的构造

例题1：有4个结点a, b, c, d，权值分别为7, 5, 2, 4，请构造一棵哈夫曼树。

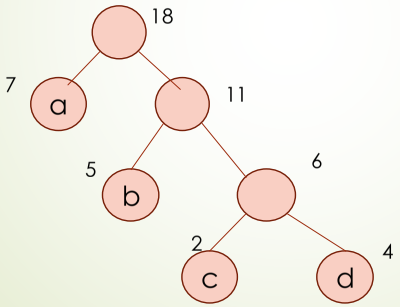
（3）从F中删除刚才选中的两棵树，同时将新得到的二叉树加入到F中，重复第2～3步。



哈夫曼树算法——哈夫曼树的构造

例题1：有4个结点a, b, c, d，权值分别为7, 5, 2, 4，请构造一棵哈夫曼树。

（3）从F中删除刚才选中的两棵树，同时将新得到的二叉树加入到F中，重复第2～3步。**直到F中只有一棵树为止。**



哈夫曼树算法——哈夫曼树的构造

练习题：

有5个叶子结点a, b, c, d, e，权值分别为2, 5, 3, 7, 1，请构造一棵哈夫曼树。

哈夫曼树算法——哈夫曼树的特点

由哈夫曼算法的基本思想可知，初始森林中共有n棵二叉树，每棵树中都仅有一个孤立的结点，它们既是根，又是叶子。

算法的第2步是，将当前森林中的两棵根结点权值最小的二叉树，合并成一棵新二叉树。每合并依次，森林中就少一棵树。

显然，要进行n-1次合并，才能使森林中的二叉树的数目由n棵减少到剩下一棵最终的哈夫曼树。

哈夫曼树算法——哈夫曼树的特点

从哈夫曼树的构造过程可以看出：

① 每个初始结点最终成为叶子结点，且权值越小的结点到根结点的路径长度越大
② 构造过程中共新建了n-1个结点（双分支结点）
③ 每次构造都选择2棵树作为新结点的孩子，因此哈夫曼树中不存在度为1的结点
④ 哈夫曼树并不唯一，但是树的带权路径长度WPL必然相同且为最优

哈夫曼树算法的实现

用一个大小为2n-1的向量来存储哈夫曼树中的结点。
存储结构为：

```
#define n        //叶子数
#define m 2*n-1    //树中结点总数
typedef struct{
  int weight;    //权值
  int plink, llink, rlink;   //双亲及左右孩子指针
}node;
node tree[m+1];
```

哈夫曼树算法的实现——框架

初始化一棵二叉树
（将tree中每个结点中的三个指针置空）

从键盘输入
（读入n个叶子的权值，分别保存着tree的前n个分量中）

构建哈夫曼树
（对森林中的树进行n-1次合并）

哈夫曼树算法的实现

初始化一棵二叉树
（将tree中每个结点中的三个指针置空）

从键盘输入
（读入n个叶子的权值，分别保存着tree的前n个分量中）

```
void InitHuffmanTree(HuffmanTree ht,int n)//初始哈夫曼树
{
  int m = 2*n-1;
  for(int i=1;i<=m;i++)
  {
    ht[i].llink=0;
    ht[i].rlink=0;
    ht[i].weight=0;
    ht[i].plink=0;
    ht[i].flag=0;
    if(i <= n)
      scanf("%d",&ht[i].weight);
  }
}
```

哈夫曼树算法的实现

构建哈夫曼树
（对森林中的树进行n-1次合并）

```
void crtHuffmanTree(HuffmanTree ht,int n)//构建哈夫曼树
{
   for(int i=n+1;i<=(2*n-1);i++)
   {
      int s1=selectmin(ht,i-1);
      int s2=selectmin(ht,i-1);
      ht[i].weight = ht[s1].weight+ht[s2].weight;
      ht[s1].plink=i;
      ht[s2].plink=i;
      ht[i].llink=s1;
      ht[i].rlink=s2;
   }
}
```

哈夫曼树算法的实现——示例

例题1：有4个结点a, b, c, d，权值分别为7, 5, 2, 4，构造一棵哈夫曼树。

初始的哈夫曼树为

结点	weight	plink	llink	rlink
1	7	0	0	0
2	5	0	0	0
3	2	0	0	0
4	4	0	0	0
5	0	0	0	0
6	0	0	0	0
7	0	0	0	0

→

结点	weight	plink	llink	rlink
1	7	0	0	0
2	5	0	0	0
3	2	5	0	0
4	4	5	0	0
5	6	0	3	4

哈夫曼树算法的实现——示例

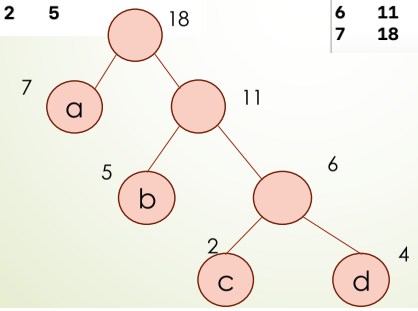
例题1：有4个结点a, b, c, d，权值分别为7, 5, 2, 4，构造一棵哈夫曼树。

结点	weight	plink	llink	rlink
1	7	0	0	0
2	5	6	0	0
3	2	5	0	0
4	4	5	0	0
5	6	6	3	4
6	11	0	2	5

→

构建后的哈夫曼树为

结点	weight	plink	llink	rlink
1	7	7	0	0
2	5	6	0	0
3	2	5	0	0
4	4	5	0	0
5	6	6	3	4
6	11	7	2	5
7	18	0	1	6



哈夫曼树的应用

1、最佳判定算法

例如，要编写一个将百分制转换成五级分制的程序。只需利用条件语句便可完成。

```
if(a < 60) b = “不及格”;
   else if(a < 70) b=“及格”;
      else if(a < 80) b=“中”;
         else if (a < 90)
b=“良”;
               else b=“优”;
```

1、最佳判定算法

例如，要编写一个将百分制转换成五级分制的程序。只需利用条件语句便可完成。

分数	0—59	60—69	70—79	80—89	90—100
比例树	0.05	0.15	0.40	0.30	0.10

```
if(a < 60) b = “不及格”;
    else if(a < 70) b=“及格”;
        else if(a < 80) b=“中”;
            else if (a < 90)
b=“良”;
                else b=“优”;
```

如果有10000条数据，那么此转换过程需要比较的次数为：

$(0.05 \times 1 + 0.15 \times 2 + 0.40 \times 3 + 0.30 \times 4 + 0.10 \times 4) \times 10000 = 31500$

```
if(a >= 90) b = “优秀”;
    else if(a >= 80) b=“良好”;
        else if(a >= 70) b=“中等”;
            else if (a >= 60) b=“及格”;
                else b=“不及格”;
```

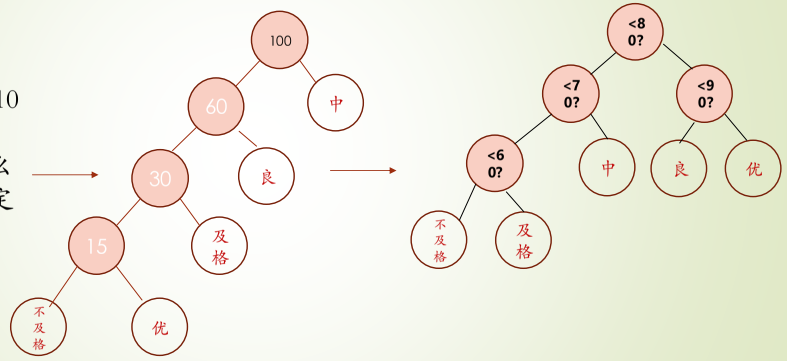
如果有10000条数据，那么此转换过程需要比较的次数为：

$(0.10 \times 1 + 0.30 \times 2 + 0.40 \times 3 + 0.15 \times 4 + 0.05 \times 4) \times 10000 = 27000$

1、最佳判定算法

分数	0—59	60—69	70—79	80—89	90—100
比例树	0.05	0.15	0.40	0.30	0.10

假定以5，15，40，30，10为权值构成一棵有5个叶子结点的哈夫曼树。那么可以得到右图所示的判定过程。



上述判定框中有两次比较，可以进一步拆分。

此时，如果有10000条数据，转换过程需要比较的次数为：

$(0.05 \times 3 + 0.15 \times 3 + 0.40 \times 2 + 0.30 \times 2 + 0.10 \times 2) \times 10000$

练习题：

假设有一组权值WG=1、3、10、15、23、35、47、66、80、99。

给出其哈夫曼树，并计算带权的路径长度。

哈夫曼树的应用

2、哈夫曼编码

哈夫曼编码是一种字符编码方式。根据字符在文件中出现的频率来建立一个用0，1串表示各字符，使平均每个字符的码长最短的最优表现形式。

在远程通讯时，需要将待传字符转换成二进制的字符串。

比如需要传送的内容为：‘ABACCDA’，这里只有4个字符，可以用两位的二进制字符串进行区分。

编码方式：

A——00

B——01

C——10

D——11

‘ABACCDA’ ⇒ 00010010101100

如果对于出现次数较多的字符，采用尽可能短的编码，可以缩减总长度

哈夫曼树的应用

2、哈夫曼编码

编码方式:
A——0
B——00
C——1
D——01

‘ABACCDA’

000011010

0000 译成什么？
AAAA?
ABA?
BB?

虽然此时传送的总长度减少到9，但是这样的内容无法实现准确译码

原因是A的编码与B的编码开始（前缀）部分相同。如果对字符进行不等长编码时，一定要求任一字符的编码都不是其他字符的前缀。

哈夫曼树的应用

2、哈夫曼编码

利用哈夫曼树设计二进制的前缀编码，可以满足内容总长最短的需要。

构造哈夫曼编码的方法:

① 统计字符集中每个字符出现的平均概率(概率越大，要求编码越短)。

② 利用哈夫曼树的特点：权越大的叶子离根越近;将每个字符的概率值作为权值，构造哈夫曼树。则概率越大的结点，路径越短

③ 在哈夫曼树的每个分支上标上0或1：结点的左分支标0，右分支标1。把从根到每个叶子的路径上的标号连接起来，作为该叶子代表的字符的编码。

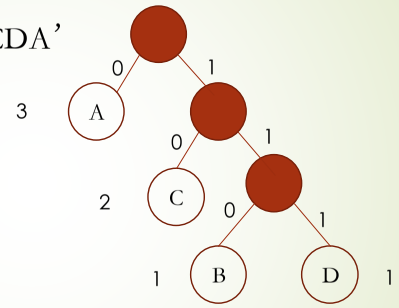
哈夫曼树的应用

2、哈夫曼编码

比如需要传送的内容为：‘ABACCDA’

哈夫曼编码方式:

A——0
C——10
B——110
D——111



哈夫曼树的应用

2、哈夫曼编码

例题：已知某系统在通信网络中只可能出现八种字符(A, B, C, D, E, F, G, H)，其出现的频率分别为0.05， 0.29，0.07，0.08，0.14，0.23，0.03，0.11。请设计哈夫曼编码。

- 根据权值W=（5，29，7，8，14，23，3，11）根据哈夫曼算法构造一棵哈夫曼树。
- 在哈夫曼树的分支上标注0，1，得到每个字符对应的编码

哈夫曼树的应用

2、哈夫曼编码

练习题：

（1）有n个叶子结点的哈夫曼树的结点总数是（ ）；

（2）有序列WG={7,19,2,6,32,3,21,10}，则所建哈夫曼树的深度为（ ），带权路径长度WPL为（ ）；

（3）在哈夫曼编码中，如果只允许编码长度小于或等于4，则除了已对两个字符编码为0和10以外，还可以最多对（ ）个字符编码。