蛟龙五班

树综合、题目选讲

Mas

#1600、Huffman树

问题描述

$Huffman$ 树在编码中有着广泛的应用。在这里，我们只关心 $Huffman$ 树的构造过程。

给出一列数 $\{p_i\}=\{p_1,p_2,\ldots,p_n\}$，用这列数构造 $Huffman$ 树的过程如下：

> *找到 $\{pi\}$ 中最小的两个数，设为 p_a 和 p_b，将 p_a 和 p_b 从 p_i 中删除掉，然后将它们的和加入到 p_i 中。这个过程的费用记为 p_a+p_b。不断重复直到 $\{p_i\}$ 中只剩下一个数。*

在上面的操作过程中，把所有的费用相加，就得到了构造 $Huffman$ 树的总费用。

对于给定的一个数列，现在请你求出用该数列构造 $Huffman$ 树的总费用。

输入格式

输入的第一行包含一个正整数 n（$n\leq 10^5$）。

接下来是 n 个正整数，表示 $p_1,p_1,\ldots,p_n$，每个数不超过 10^9。

输出格式

输出用这些数构造 $Huffman$ 树的总费用。

样例解释

找到 $5,3,8,2,9$ 中最小的两个数，分别是 2 和 3，从 $\{p_i\}$ 中删除它们并将和 5 加入，得到 $\{5,8,9,5\}$，费用为 5。

找到 $5,8,9,5$ 中最小的两个数，分别是 5 和 5，从 $\{p_i\}$ 中删除它们并将和 10 加入，得到 $\{8,9,10\}$，费用为 10。

找到 $8,9,10$ 中最小的两个数，分别是 8 和 9，从 $\{p_i\}$ 中删除它们并将和 17 加入，得到 $\{10,17\}$，费用为 17。

找到 $10,17$ 中最小的两个数，分别是 10 和 17，从 $\{p_i\}$ 中删除它们并将和 27 加入，得到 $\{27\}$，费用为 27。

现在，数列中只剩下一个数 27，构造过程结束，总费用为 $5+10+17+27=59$。

样例输入

```
5
5 3 8 2 9
```

样例输出

```
59
```

#1600、Huffman树

设二叉树有n个带权叶结点,从根到各叶结点的路径长度与叶节点权值的乘积之和称为树的带权路径长度$(Weighted\ Path\ Length\ of\ Tree)$

设w_i为二叉树地i个叶结点的权值,l_i为从根结点到第i个叶结点的路径长度,则 WPL 计算公式如下:

$$WPL = \sum_{i=1}^{n} w_i \times l_i$$

对于给定一组具有确定权值的叶结点,可以构造出不同的二叉树,其中WPL最小的二叉树 称为霍夫曼树$(Huffman\ Tree)$

对于霍夫曼树来说,其叶结点权值越小离根越远,叶结点权值越大离根越近,此外其仅有叶结点的度为0,其他结点度均为2

霍夫曼算法用于构造一棵霍夫曼树:

1. 由给定的n个权值构造n棵只有一个根节点的二叉树,得到一个二叉树集合F
2. 从集合F中选取权值最小的两棵二叉树作为左右子树构造一棵新二叉树,新二叉树的根节点的权值为左、右子树根结点的权值和
3. 从F中删除作为左、右子树的两棵二叉树,并将新建立的二叉树加入到F中
4. 重复 2、3 步,当集合中只剩下一棵二叉树时,这棵二叉树就是霍夫曼树

#1600、Huffman树

通常给每一个字符标记一个单独的代码来表示一组字符,即编码

在进行二进制编码时,假设所有的代码都等长,表示 n 个不同的字符需要 $\lceil \log_2 n \rceil$位,称为等长编码

若各字符的使用频率相等,那么等长编码是空间效率最高的编码方法

但字符出现的频率不同,可让频率高的字符采用尽可能短的编码,频率低的字符采用尽可能长的编码,构造不等长编码,获得更好的空间效率

在设计不等长编码时,要考虑解码的唯一性

若一组编码中任一编码都不是其他任何一个编码的前缀,那么称这组编码为前缀编码,其保证了编码被解码时的唯一性

霍夫曼树可用于构造 最短的前缀编码,即 霍夫曼编码($Huffman\ Code$),其构造步骤如下

1. 设需要编码的字符集为: $d_1, d_2, \dots, d_n$,出现的频率为: $w_1, w_2, \dots, w_n$
2. 以 $d_1, d_2, \dots, d_n$作为叶结点,$w_1, w_2, \dots, w_n$作为叶结点的权值,构造一棵霍夫曼树
3. 规定哈夫曼编码树的左分支代表0,右分支代表1,则从根结点到每个叶结点所经过的路径组成的 0、1 序列即为该叶结点对应字符的编码

#2114、表达式树

题目描述

给定一个语法二叉树，请你输出相应的中缀表达式，并利用括号反映运算符的优先级。

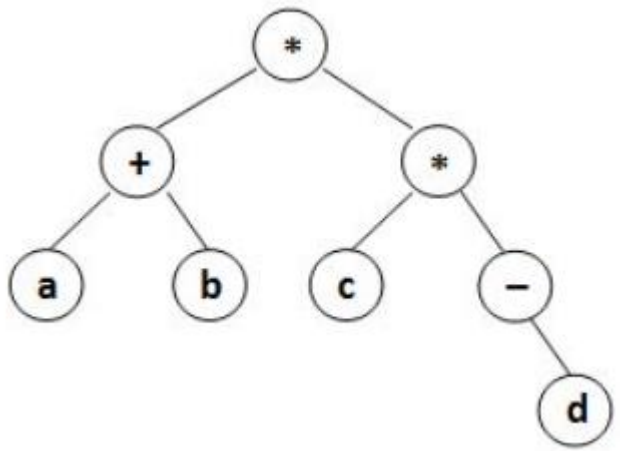
输入格式

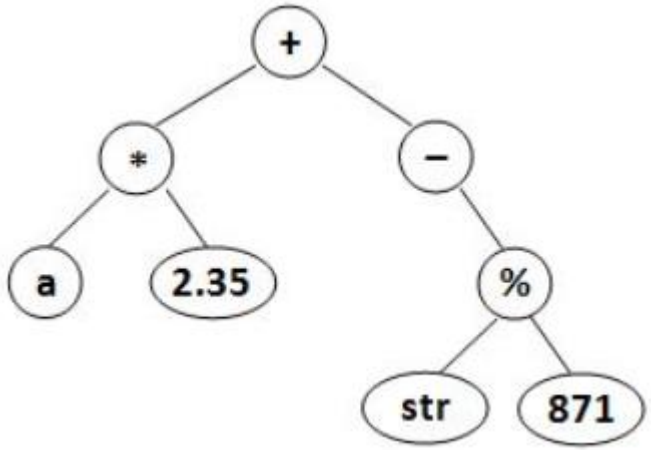
第一行包含整数 N 表示二叉树的总结点个数。

接下来 N 行，每行以下列格式给出一个结点的信息（第 i 行对应于第 i 个结点）：`data left_child right_child` 其中 `data` 是一个长度不超过 `10` 的字符串，`left_child` 和 `right_child` 分别是该结点的左右子结点编号。

所有结点编号从 1 到 N，$NULL$ 用 -1 表示。

以下两个图分别对应样例 1 和样例 2。





输出格式

请在一行输出中缀表达式，并利用括号反映运算符的优先级。

注意，不能有多余括号，请任何符号之间不得有空格。

输入样例1：

```
8
* 8 7
a -1 -1
* 4 1
+ 2 5
b -1 -1
d -1 -1
- -1 6
c -1 -1
```

输入样例2：

```
8
2.35 -1 -1
* 6 1
- -1 4
% 7 8
+ 2 3
a -1 -1
str -1 -1
871 -1 -1
```

输出样例1：

```
(a+b)*(c*(-d))
```

输出样例2：

```
(a*2.35)+(-(str%871))
```

数据范围

对于全部的数据 $1 \leq N \leq 20$

#2114、表达式树

表达式树的前中后序遍历对应表达式的前中后缀表达式

将树存储后,分别进行中序遍历

叶子节点和根节点输出时不使用括号包裹

其他节点使用括号包裹

```
void inOrder(int root, int deep)
{
  if (root == -1)
    return;
  if (deep > 1 && (tree[root].l != -1 || tree[root].r != -1))
    cout << "(";
  inOrder(tree[root].l, deep + 1);
  cout << tree[root].data;
  inOrder(tree[root].r, deep + 1);
  if (deep > 1 && (tree[root].l != -1 || tree[root].r != -1))
    cout << ")";
}
```

二叉搜索树

二叉搜索树是一种二叉树的树形数据结构，其定义如下：

- 空树是二叉搜索树
- 若二叉搜索树的左子树不为空,则其左子树上所有点的附加权值均小于其根节点的值
- 若二叉搜索树的右子树不为空则其右子树上所有点的附加权值均大于其根节点的值
- 二叉搜索树的左右子树均为二叉搜索树

二叉搜索树上的基本操作所花费的时间与这棵树的高度成正比

对于一个有n个结点的二叉搜索树中,这些操作的最优时间复杂度为$O(\log n)$,最坏为$O(n)$

随机构造这样一棵二叉搜索树的期望高度为 $O(\log n)$

由二叉搜索树的递归定义可得,二叉搜索树的中序遍历权值的序列为非降的序列

#2194、二叉搜索树的2层结点统计

题目描述

二叉搜索树或者是一棵空树，或者是具有下列性质的二叉树：

- 若它的左子树不空，则左子树上所有结点的值均小于或等于它的根结点的值
- 若它的右子树不空，则右子树上所有结点的值均大于它的根结点的值
- 它的左、右子树也分别为二叉搜索树

将一系列数字按给定顺序插入一棵初始为空的二叉搜索树,你的任务是统计结果树中最下面 2 层的结点数

输入格式

输入在第一行给出一个正整数 $N(1 \leq N \leq 1000)$,为插入数字的个数第二行给出 N 个 $[-1000, 1000]$ 区间内的整数。数字间以空格分隔

输出格式

在一行中输出最下面 2 层的结点总数

输入样例

```
9
25 30 42 16 20 20 35 -5 28
```

输出样例

```
6
```

#2194、二叉搜索树的2层结点统计

定义 $insert(root, val)$ 为在以$root$为根节点的二叉搜索树中插入一个值为val的新节点

分类讨论

- 若$root$为空,直接返回一个值为val的新节点
- 若$root$的权值大于等于val,在 的左子树中插入权值为val的节点
- 若$root$的权值小于val ,在 $root$的右子树中插入权值为val的节点

```
int insert(int root, int val)
{
  if (root == -1)
  {
    tree[++idx].val = val;
    return idx;
  }
  if (val <= tree[root].val)
    tree[root].l = insert(tree[root].l, val);
  else
    tree[root].r = insert(tree[root].r, val);
  return root;
}
```

#1783、填充二叉搜索树

题目描述

二叉搜索树（BST）递归定义为具有以下属性的二叉树：

- 若它的左子树不空，则左子树上所有结点的值均小于它的根结点的值
- 若它的右子树不空，则右子树上所有结点的值均大于或等于它的根结点的值
- 它的左、右子树也分别为二叉搜索树

给定二叉树的具体结构以及一系列不同的整数，只有一种方法可以将这些数填充到树中，以使结果树满足 BST 的定义。

请你输出结果树的层序遍历。

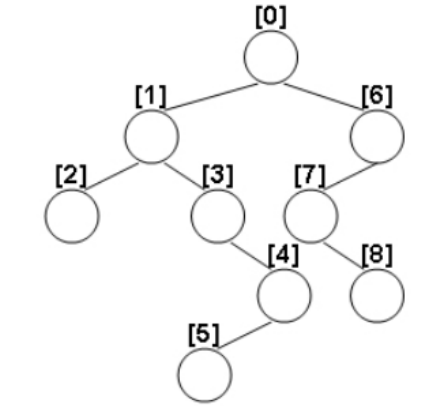


Figure 1

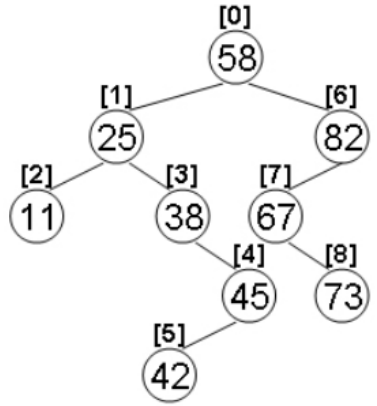


Figure 2

数据范围

对于全部的数据 $1 \le N \le 100$

输入格式

第一行包含一个正整数 N，表示树的结点个数。

所有结点的编号为 $0 \sim N-1$，并且编号为 0 的结点是根结点。

接下来 N 行，第 i 行（从 0 计数）包含结点 i 的左右子结点编号。如果该结点的某个子结点不存在，则用 `-1` 表示。

最后一行，包含 N 个不同的整数，表示要插入树中的数值。

输出格式

输出结果树的层序遍历序列。

输入样例：

```
9
1 6
2 3
-1 -1
-1 4
5 -1
-1 -1
7 -1
-1 8
-1 -1
73 45 11 58 82 25 67 38 42
```

输出样例：

```
58 25 82 11 38 67 45 73 42
```

#1783、填充二叉搜索树

二叉搜索树中序遍历结果为有序序列

将数组排序,对树进行中序遍历

- 先递归填充左子树
- 再填充当前节点
- 最后递归填充右子树

最后对树进行遍历输出

```
void build(int u)
{
  if (u == -1)
    return;
  build(tree[u].l);
  tree[u].data = a[idx++];
  build(tree[u].r);
}
```

谢谢观看