实验四

1. **问题描述与实验目的：**

给定 n 个字母（或字）在文档中出现的频率序列 X=<x1,x2,…,xn>，求出这 n 个字母的 Huffman 编码。为方便起见，以下将频率用字母出现的次数（或称权值）w1,w2,…,wn 代替。

2. **要点分析和算法描述：**

先构造哈夫曼树：用优先队列构造最小堆，将所有孤立节点放入队列。每次从队列中取出两个权值最小的节点，将它们合并，并按照规则，将权值较大的节点放在左边，产生的新节点加入最小堆，新节点权值为两个子节点权值之和。

再进行哈夫曼编码操作：使用 dfs 方式，从根结点开始，依次遍历左右子树，左子树哈夫曼编码加 0，右子树哈夫曼编码加 1。

3. **代码实现：**

```cpp
#include <iostream>
#include <vector>
#include <queue>
using namespace std;

// 节点结构体
struct node
{
    int l, r, w;
    int id;
    string code;
    bool operator<(const node &a) const
    {
        if (w == a.w)
            return id < a.id; // 先输入权值的后处理
        return w > a.w;       // 构造最小堆
    }
};

class tree
{
public:
    int cnt; // 记录当前节点 id
    int n;   // 开辟空间大小，2*num 数量
    node *head;

    tree(int num) : cnt(0), n(2 * num)
    {
        head = new node[n];
    };
    // 插入操作，左右孩子、权值、id
```

```
    void insert(int lchild, int rchild, int weight)
    {
        head[cnt].l = lchild;
        head[cnt].r = rchild;
        head[cnt].w = weight;
        head[cnt].id = cnt;
        cnt++;
    }
    // 合并两个节点，并将新节点加到head 中
    void union_node(int a, int b)
    {
        insert(a, b, head[a].w + head[b].w);
    }

    ~tree()
    {
        delete[] head;
    }
    // 生成哈夫曼编码
    void dfs(int u, string s)
    {
        if (u == -1)
            return;
        head[u].code = s;        // 保存编码
        dfs(head[u].l, s + "0"); // 左子树编码为0
        dfs(head[u].r, s + "1"); // 右子树编码为1
    }
    // 构造哈夫曼树
    void solve()
    {
        // 最小堆，按照w 从小到大排序
        priority_queue<node> q;
        for (int i = 0; i < cnt; i++)
            q.push(head[i]);
        // 构造哈夫曼树，每次取出两个最小的节点合并
        while (q.size() > 1)
        {
            //a 为权值最小的节点，b 为权值次小的节点
            node a = q.top();
            q.pop();
            node b = q.top();
            q.pop();
            union_node(b.id, a.id); // 只添加了合并的非叶子节点，将较大节点放哈
夫曼树中左边
```

```
            q.push(head[cnt - 1]);
        }
        //最终队列中只剩下一个根节点
        int c = q.top().id;
        dfs(c, "");         // 从根节点开始dfs
    }
};

int main()
{
    ios::sync_with_stdio(0);
    int T;
    cin >> T;
    int k = 1;
    while (k <= T)
    {
        int n;
        cin >> n;
        tree tr(n + 1);
        // 将n个点放入head数组
        for (int i = 1; i <= n; i++)
        {
            int x;
            cin >> x;
            // 当前每个节点为孤立节点
            tr.insert(-1, -1, x);
        }
        // 构造哈夫曼树
        tr.solve();
        cout << "Case " << k++ << endl;
        // 输出哈夫曼节点权值和编码
        for (int i = 0; i < n; i++)
            cout << tr.head[i].w << " " << tr.head[i].code << endl;
    }
    return 0;
}
```

4. 运行结果

```
2
6
9 8 3 4 1 2
Case 1
9 00
8 01
3 100
4 11
1 1011
2 1010
8
60 20 5 5 3 3 3 1
Case 2
60 0
20 10
5 1101
5 1110
3 11000
3 11001
3 11110
1 11111
```

5. 实验体会

本题在用节点构造最小堆时注意：默认的 priority_queue 使用的是“<”比较符号，需要重载运算符“<”,return a.w > b.w。

在构造哈夫曼树时注意：权值较大的节点在左边，因此在取出优先队列头两个节点后，a = q.top(); q.pop();b = q.top(); q.pop();需要将 b 放到左边，这点细节需要注意。

最后找根结点，进行 dfs 构造哈夫曼编码时，有个小技巧，剩在队列中的最后一个节点即为根结点，取出即可。