**算法实习报告：实验三 最优合并问题**

题目： 最优合并问题

1. **需求分析**

给定k个排好序的序列s1,s2,...,sk，用2路合并算法将这k个序列合并成一个序列。假设所采用的2路合并算法合并2个长度为m和n的序列需要m+n-1次比较。设计一个算法确定2路合并次序，使所需的总比较次数最少。

1. **概要设计**

要总比较次数最少，只需要每次选择两个最短的序列合并。可以使用哈弗曼树进行求解。

将序列长度作为哈弗曼树的带权路径长度构造一棵哈弗曼树，根据具有最小带权路径长度的哈弗曼树根节点上的数求最少的比较次数。

1. **详细设计**

#include "iostream"

using namespace *std*;

//存储Huffman树

typedef struct {

int weight;

int parent;

int lchild;

int rchild;

}HTNode, \*HuffmanTree;

void createHuffmanTree(HuffmanTree ht, int n);

void Select(HuffmanTree ht, int k, int &min1, int &min2);

int min(HuffmanTree ht, int k);

//创建Huffman树

void createHuffmanTree(HuffmanTree ht, int n) {

int compTime = 0;

//初始化

for (int i = 0; i < 2 \* n - 1; i++) {

ht[i].parent = -1;

ht[i].lchild = -1;

ht[i].rchild = -1;

}

int min1 , min2;

for (int i = n; i < 2 \* n - 1; i++) {

Select(ht, i, min1, min2); //选择出没有双亲且权值最小的两个节点的序列

ht[min1].parent = i;

ht[min2].parent = i;

ht[i].lchild = min1;

ht[i].rchild = min2;

ht[i].weight = ht[min1].weight + ht[min2].weight;

compTime += ht[i].weight - 1;

}

*cout* << compTime << *endl*; //输出中的比较次数

}

//查找权值最小的两个点

void Select(HuffmanTree ht, int k, int &min1, int &min2) {

min1 = min(ht, k);

min2 = min(ht, k);

}

//查找权值最小的一个点

int min(HuffmanTree ht, int k) {

int i = 0;

int minIndex;

int minWeight;

while (ht[i].parent != -1) {

i++;

}

minIndex = i;

minWeight = ht[minIndex].weight;

for (; i < k; i++) {

if (ht[i].weight < minWeight && ht[i].parent == -1) {

minIndex = i;

minWeight = ht[minIndex].weight;

}

}

ht[minIndex].parent = -2;

return minIndex;

}

int main() {

int n;

*cin* >> n;

HTNode ht[100];

for (int i = 0; i < n; i++) {

*cin* >> ht[i].weight;

}

createHuffmanTree(ht, n);

return 0;

}

1. **调试分析**

使用Huffman树，进行查询操作，该算法的时间复杂度是O(nlogn)。

1. **用户手册**

1、 数据输入：首先输入序列的个数k。之后输入k个序列各自的长度。

2、 输出：最小的总比较次数。

1. **测试结果**

测试了几组数据终端输出结果如下：

1. 第一组

输入：

k 为 3。

k 组序列的长度各为 4, 10, 6。

输出：

最少比较次数为 28。



1. 第二组

输入：

k 为 4。

k 组序列的长度各为 5, 12, 11, 2。

输出：

最少比较次数为 52。

