浙江工业大学

算法分析与设计实验报告

(2021 级)

实验题目	实验 1
学生姓名	温家伟
学生学号	202103151422
专业班级	大数据分析 2101
所在学院	理学院
提交日期	2023-3-8

实验目的

掌握贪心算法的解题步骤

第四章 贪心算法

实验内容

在北美洲东南部，有一片神秘的海域，那里碧海 蓝天、阳光明媚，这正是传说中海盗最活跃的加勒比 海（Caribbean Sea）。17 世纪时，这里更是欧洲大陆 的商旅舰队到达美洲的必经之地，所以当时的海盗活 动非常猖獗，海盗不仅攻击过往商人，甚至攻击英国 皇家舰……
有一天，海盗们截获了一艘装满各种各样古董的 货船，每一件古董都价值连城，一旦打碎就失去了它 的价值。虽然海盗船足够大，但载重量为 C，每件古 董的重量为 wi，海盗们该如何把尽可能多数量的宝贝 装上海盗船呢?

输入格式:

第1行输入T组测试数据，每组测试数据输入载重量 c 及古董个数 n，下1行输入每个古董的重量wi，用空格分开.

输出格式:

每组能装入的古董最大数量

输入样例:

```
1
30 8
4 10 7 11 3 5 14 2
```

输出样例:

```
5
```

7-2 活动选择问题 分数 25

全屏浏览题目 切换布局 作者 李廷元 单位 中国民用航空飞行学院

假定一个有n个活动(activity)的集合S={a_1,a_2,....,a_n}，这些活动使用同一个资源（例如同一个阶梯教室），而这个资源在某个时刻只能供一个活动使用。每个活动a_i都有一个开始时间s_i和一个结束时间f_i，其中$0 <= s_i < f_i <= 32767$。如果被选中，任务a_i发生在半开时间区间$[s_i, f_i)$期间。如果两个活动a_i和a_j满足$[s_i, f_i)$和$[s_j, f_j)$不重叠，则称它们是兼容的。也就说，若$s_i >= f_j$或$s_j >= f_i$，则a_i和a_j是兼容的。在活动选择问题中，我们希望选出一个最大兼容活动集。

输入格式:

第一行一个整数n(n≤1000);

接下来的n行，每行两个整数，第一个s_i，第二个是$f_i(0 <= s_i < f_i <= 32767)$。

输出格式:

输出最多能安排的活动个数。

输入样例:

```
11
3 5
```

7-3 最小生成树-kruskal 分数 10

全屏浏览题目 切换布局 作者 任唯 单位 河北农业大学

题目给出一个无向连通图，要求求出其最小生成树的权值。

温馨提示：本题请使用kruskal最小生成树算法。

输入格式:

第一行包含两个整数 $N(1 <= N <= 1x10^6), M(1 <= M <= 1x10^6)$ 表示该图共有 N 个结点和 M 条无向边。

接下来 M 行每行包含三个整数 X_i, Y_i, Z_i ，表示有一条长度为 Z_i 的无向边连接结点 X_i, Y_i 。

输出格式:

输出一个整数表示最小生成树的各边的长度之和。

输入样例:

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

题目描述:

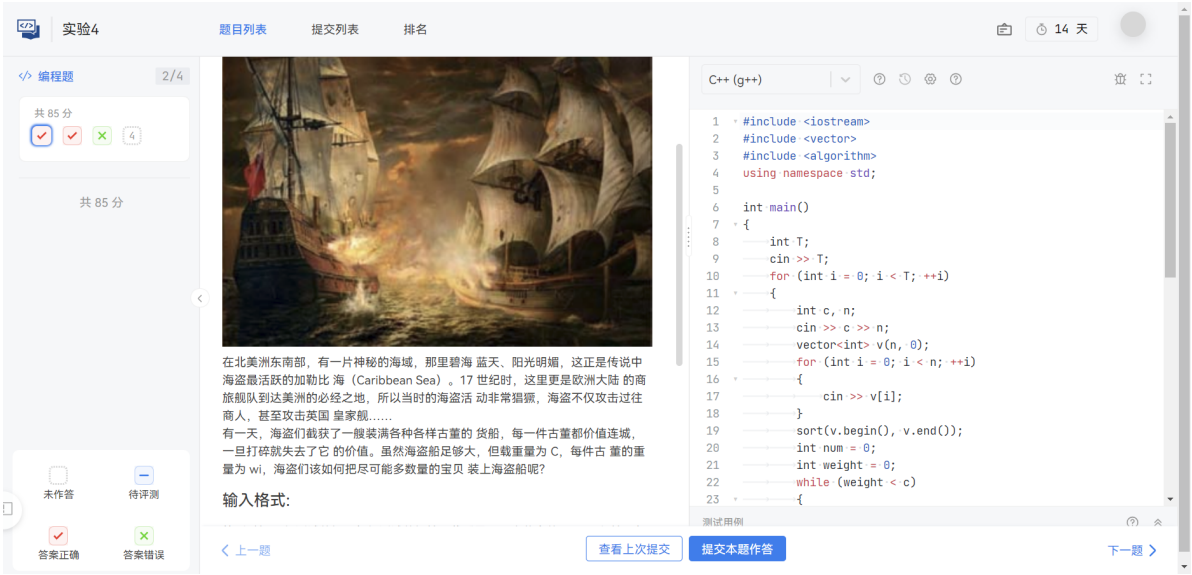
哈夫曼树(Huffman Tree)又称最优二叉树，是一种带权路径长度最短的二叉树。所谓树的带权路径长度，就是树中所有的叶结点的权值乘上其到根结点的路径长度（若根结点为0层，叶结点到根结点的路径长度为叶结点的层数）。树的路径长度是从树根到每一结点的路径长度之和，记为
WPL=（W1*L1+W2*L2+W3*L3+...+Wn*Ln），N个权值Wi（i=1,2,...n）构成一棵有N个叶结点的二叉树，相应的叶结点的路径长度为Li（i=1,2,...n）。可以证明哈夫曼树的WPL是最小的。

在数据通信中，需要将传送的文字转换成二进制的字符串，用0，1码的不同排列来表示字符。例如，需传送的报文为“AFTER DATA EAR ARE ART AREA”，这里用到的字符集为“A，E，R，T，F，D”，各字母出现的次数为{8，4，5，3，1，1}。现要求为这些字母设计编码。要区别6个字母，最简单的二进制编码方式是等长编码，固定采用3位二进制，可分别用000、001、010、011、100、101对“A，E，R，T，F，D”进行编码发送，当对方接收报文时再按照三位一分进行译码。显然编码的长度取决报文中不同字符的个数。若报文中可能出现26个不同字符，则固定编码长度为5。然而，传送报文时总是希望总长度尽可能短。在实际应用中，各个字符的出现频度或使用次数是不相同的，如A、B、C的使用频率远远高于X、Y、Z，自然会想到设计编码时，让使用频率高的用短码，使用频率低的用长码，以优化整个报文编码

实验结果及相应代码

1.加勒比海盗船——最优装载问题

1.1 PTA提交代码截图



1.2 PTA提交代码结果截图

1.3 算法分析

> 把尽可能多的宝贝装上海盗船，贪心策略是先挑重量轻的宝贝，所以排个序即可。

2.活动选择问题

2.1 PTA提交代码截图

2.2 PTA提交代码结果截图

2.3 算法分析

> 活动选择问题，想要尽可能多的安排活动，就应该先让结束时间早的来安排，所以排序的比较规则是按第二个整数来的。

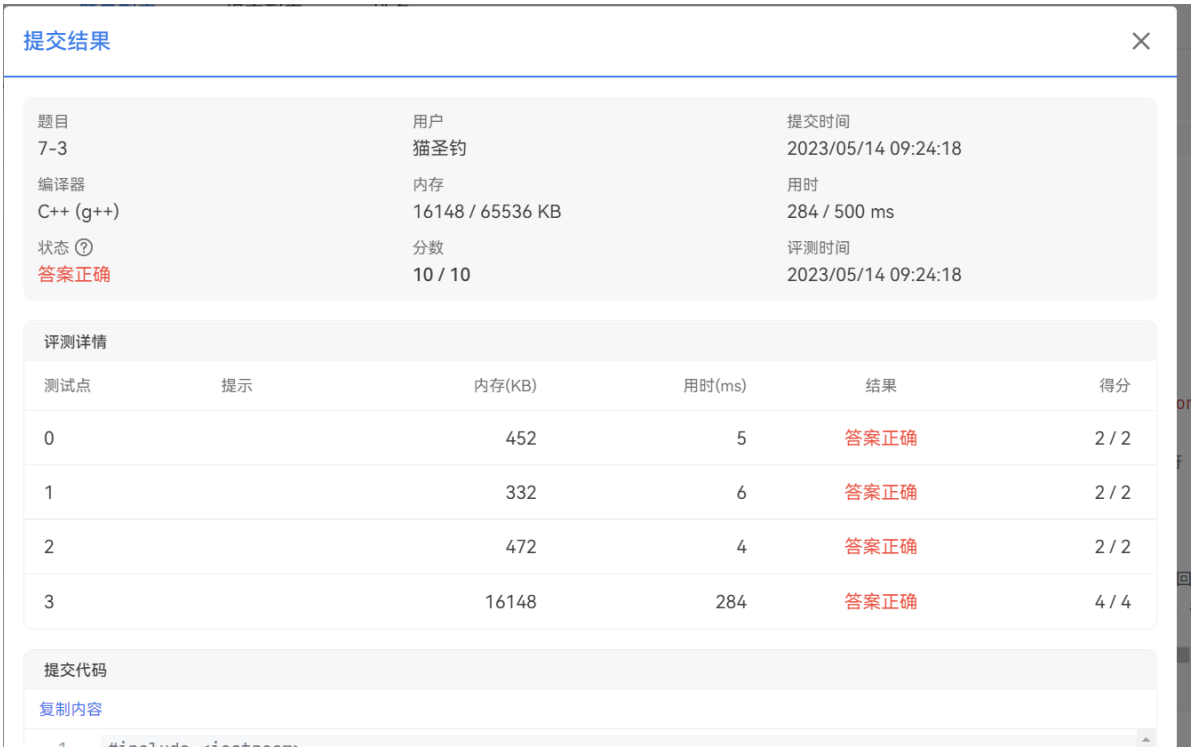
3 最小生成树-kruskal

3.1 PTA提交代码截图

```
const int N = 1000001;
int p[N];//保存并查集
int n,m;
struct Edge
{
    int a,b,w;//两个顶点和一个权值
    bool operator<(const Edge &W)const
    {
        return w<W.w;//重载比较运算符
    }
}edges[N*2];
int res = 0;
int cnt = 0;//用来计算已经连通的不形成回路的边的数量
int find(int a)//递归查找元素的父节点，并将途中的所有元素的父节点
{
    if(p[a]!=a) p[a] = find(p[a]);
    return p[a];
}
void kruskal()
{
    for(int i=0;i<m;i++)
    {
        int pa = find(edges[i].a);//a所在的集合
        int pb = find(edges[i].b);//b所在的集合
        if(pa!=pb)//如果a,b不在一个集合中
        {
            res+=edges[i].w;//a,b形成的这条边符合题意
            p[pa] = pb;//合并a,b
            cnt++;//符合题意的边的数量+1
        }
    }
}
int main()
{
    scanf("%d%d",&n,&m);
    for(int i=0;i<n;i++) p[i] = i;//初始化并查集
    for(int i=0;i<m;i++)//读入每条边
```

```
    {
        int a,b,w;
        scanf("%d%d%d",&a,&b,&w);
        edges[i] = {a,b,w};
    }
    sort(edges,edges+m);//按照边长
    kruskal();
    if(cnt<n-1)
    {
        cout<<"impossible";
        return 0;
    }
    printf("%d",res);//输出权重之和
    return 0;
}
```

3.2 PTA提交代码结果截图



3.3 算法分析

> (Prim 算法) O(n^2)
>
> Prim 算法其实和 Dijkstra 算法很像，都是每次找离当前集合最近的点，不同的是 Dijkstra 只要找到的点不在集合中就加入了，而 Prim 要加入集合。因此 Prim 算法的复杂度是 O(n^2)，Dijkstra 算法是 O(nm)。

5 构造哈夫曼树-有序输入

5.1 PTA提交代码截图

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

typedef struct Haffm {
    int priority;//权重
    Haffm* lchild;
    Haffm* rchild;
}Haffmtree;

void midorder(Haffmtree*& root) {//中序遍历打印结果
    if (root==NULL)
        return;
    midorder(root->lchild);
    cout << root->priority <<" ";
    midorder(root->rchild);
}
struct myCompare {//比较函数，小的在前
    bool operator()(Haffmtree*& a, Haffmtree*& b) {
        return a->priority > b->priority;
    }
};

Haffmtree* buildHaffmtree(priority_queue<Haffmtree*, vect
{
    do {
        Haffmtree* first = haffm.top();
        haffm.pop();
        if (haffm.empty()) {
            return first;
        }//哈夫曼数已经完成排序，first是根结点
        Haffmtree* root = new Haffmtree;
        root->lchild = first;
        root->rchild = haffm.top();
        haffm.pop();
        root->priority = root->lchild->priority + root->r
        //根结点权重是左右子树权重之和。
        haffm.push(root);//将根结点入队列
    } while (!haffm.empty());
}
int main() {
```

```
42      priority_queue<Haffmtree*, vector<Haffmtree*>, myCompa
43      int n;
44      cin >> n;
45      while (n--) {
46          Haffmtree* node = new Haffmtree;
47          cin >> node->priority;
48          node->lchild = node->rchild = nullptr;
49          haffm.push(node);
50      }
51      Haffmtree* root = buildHaffmtree(haffm);
52      midorder(root);
53      return 0;
54  }
55
56
57
```

5.2 PTA提交代码结果截图

提交结果

题目	用户	提交时间
7-5	猫圣钓	2023/05/14 10:24:13
编译器	内存	用时
C++ (g++)	592 / 65536 KB	6 / 400 ms
状态	分数	评测时间
答案正确	15 / 15	2023/05/14 10:24:13

评测详情

测试点	提示	内存(KB)	用时(ms)	结果	得分
test		592	5	答案正确	5 / 5
test2		324	4	答案正确	5 / 5
test3		472	6	答案正确	5 / 5

提交代码

复制内容

5.3 算法分析

> **思路：**
>
> 1. 建立一个元素是二叉树结点的优先队列，要求权重小的在前。

> 2. 将输入的元素入队列。
> 3. 反复取出前两个结点，组成新结点，然后入队列，直到队列为空。此时队列的最后一个结点就是哈夫曼二叉树的根结点。
> 4. 中序遍历二叉树。