数据结构上机实验题报告

题目：哈夫曼树

姓名：

班级：

学号：

提交日期2021.11.28

**一、题目**

描述题目所要求的功能，说明输入和输出。

1. 统计数据中各个字符出现的频率存储到一个特定的数组中
2. 依据字符频率构建最优化字符编码的哈夫曼树，对一个接收数据进行解码

**二、程序设计**

描述数据类型定义，或抽象数据类型定义。

struct huffmanNode{

char c;

int w,tag,left,right;

}node[MAX\*2+10]={0};

使用线性表来储存哈夫曼节点。

一个哈夫曼节点中包括了当前的存储的char值c和对应的频率值w，而tag是指示这个c是否出现过的标志。left和right是两个子节点在线性表中对应的位置。

描述算法执行步骤，用类C语言描述算法。

第一、初始化数据，输入数据

第二、依据数据构建二叉树（数据都是ASCII码）

第三、接受二进制数，对二进制数解码

**注意：（这里用的循环，当输入0或1时，程序会将输入的数当二进制数，当输入q时程序退出循环，当输入其他时程序无反应）**

分析算法的时间复杂度和空间复杂度。

时间复杂度 O(n2)，瓶颈是构建哈夫曼树

空间复杂度 O(1)，主要是排序节点

**三、程序测试运行**

说明编译环境和运行环境

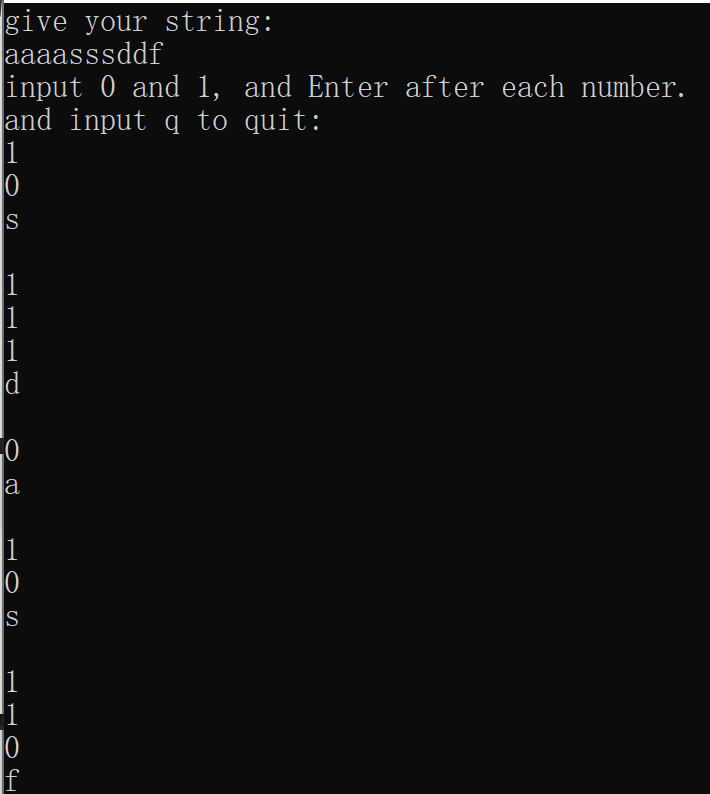
Codeblocks

说明测试数据，输入的数据与输出的测试结果。

输入的串为aaaasssddf

经过哈夫曼编码，a-0 s-10 d-111 f-110

检验如下：



**四、编程工作总结**

说明编程工作内容、心得，讨论哪些可以改进。

**内容：**对于要求的多个需求，抽象出输入、构建、解码等三个主要工作流程，分步逐渐实现功能。详情见源代码

**心得：**

1. 使用线性表来表示哈夫曼树，因为储存的节点数是有限的，并且这样的对应关系比较好
2. 利用各个字符的ascii码大小关系进行编码，更有条理

**改进：**

1. 构建哈夫曼树时可以使用优先队列，时间复杂度降低为O（logn）

**五、程序源代码**

1. #include<stdio.h>
2. #include<string.h>
3. #define MAX 128
4. struct huffmanNode{
5. char c;
6. int w,tag,left,right;
7. }node[MAX\*2+10]={0};
8. int cnt=0,ch[MAX]={0};
9. int main(){
10. *// 数据初始化：*
11. char str[1000],k;
12. int i=0,j,temp;
13. printf("give your string:\n");
14. scanf("%s",str);
15. while(str[i])  ch[(int)str[i++]]++;
16. for(i=0;i<MAX;i++){
17. if(ch[i]!=0){
18. node[cnt].c=(char)(i);
19. node[cnt].w=ch[i];
20. node[cnt++].tag=0;
21. }
22. }
23. *// 构建霍夫曼树*
24. int min1,min2,x1,x2;
25. for(i=0;i<cnt-1;i++){
26. min1=0xffff;min2=0xffff;
27. x1=-1;x2=-1;
28. for(j=0;j<cnt+i;j++){
29. if(node[j].w<min1&&node[j].tag==0){
30. min2=min1;x2=x1;
31. min1=node[j].w;x1=j;
32. }
33. else if(node[j].w<min2&&node[j].tag==0){
34. min2=node[j].w;x2=j;
35. }
36. }
37. node[x1].tag=1;node[x2].tag=1;
38. node[cnt+i].w=min1+min2;
39. node[cnt+i].left=x1;node[cnt+i].right=x2;*// 在此霍夫曼树中，树的右节点的权值大一些。*
40. node[cnt+i].tag=0;
41. }
42. printf("input 0 and 1, and Enter after each number.\nand input q to quit:\n");
43. *// 解码：*
44. k=getchar();
45. while(k!='q'){
46. temp=2\*cnt-2;
47. while(k!='0'&&k!='1'&&k!='q')k=getchar();
48. while(temp>=cnt){
49. if(k=='1'){
50. temp=node[temp].right;
51. }
52. else if(k=='0'){
53. temp=node[temp].left;
54. }
55. else if(k=='q'){
56. break;
57. }
58. k = getchar();
59. }
60. if(k!='q')
61. printf("%c\n\n",node[temp].c);
62. }
63. return 0;
64. }