LESSON_10 参考试题

一、 编程题

1. 图像压缩（2023年6月）

【问题描述】

图像是由很多的像素点组成的。如果用 0 表示黑，255 表示白，0 和 255 之间的值代表不同程度的灰色，则可以用一个字节表达一个像素（取值范围为十进制 0-255、十六进制 00-FF）。这样的像素组成的图像，称为 **256 级灰阶**的灰度图像。

0 **255**

现在希望将 256 级灰阶的灰度图像压缩为 16 级灰阶，即每个像素的取值范围为十进制 0-15、十六进制 0-F。**压缩规则为：**

① 统计出每种灰阶的数量，取数量**最多的前 16 种灰阶**（如某种灰阶的数量与另外一种灰阶的数量相同，则以灰阶值**从小到大**为序），分别编号 **0-F（**最多的编号为 0，以此类推）。

② 其他灰阶转换到**最近的** 16 种灰阶之一，将某个点灰阶数与 16 种灰阶中的一种相减，绝对值**最小**即为最近，如果绝对值相等，则编号较小的灰阶更近。

【输入描述】

输入第 1 行为一个正整数 N，表示接下来有 N 行数据组成一副 256 级灰阶的灰度图像。约定 **10≤N≤20**。

第 2 行开始的行，每行为长度相等且为**偶数**的**字符串**，每**两个字符**用十六进制表示一个像素。约定输入的灰度图像至少有 16 种灰阶。约定每行最多 20 个像素。

【输出描述】

第一行输出压缩选定的 16 种灰阶的十六进制编码，共计 32 个字符。

第二行开始的行，输出压缩后的图像，每个像素一位十六进制数表示压缩后的灰阶值。

【样例输入】

10
00FFCFAB00FFAC09071B5CCFAB76
00AFCBAB11FFAB09981D34CFAF56
01BFCEAB00FFAC0907F25FCFBA65
10FBCBAB11FFAB09981DF4CFCA67
00FFCBFB00FFAC0907A25CCFFC76
00FFCBAB1CFFCB09FC1AC4CFCF67
01FCCBAB00FFAC0F071A54CFBA65
10EFCBAB11FFAB09981B34CFCF67
01FFCBAB00FFAC0F071054CFAC76
1000CBAB11FFAB0A981B84CFCF66

【样例输出】

ABCFFF00CB09AC07101198011B6776FC
321032657CD10E
36409205ACC16D
B41032657FD16D
8F409205ACF14D
324F326570D1FE
3240C245FC411D
BF4032687CD16D
8F409205ACC11D
B240326878D16E
83409205ACE11D

【样例解释】

灰阶'AB'、'CF'和'FF'出现 14 次，'00'出现 10 次，'CB'出现 9 次，'09'出现 7 次，'AC'出现 6 次，'07'出现 5 次，'10'、'11'和'98'出现 4 次，'01'、'1B'、'67'、'76'和'FC'出现 3 次。

【参考代码】

```
#include <bits/stdc++.h>
using namespace std;
// 一位十六进制字符转换为数字
int trans(char a) {
    if (a<='9')
        return (a-'0');
    return (a-'A'+10);
}
// 一位十六进制数字转换为字符
char itrans(int n) {
    if (n>=10)
        return (char)(n-10+'A');
    return (char)(n+'0');
}
int image[25][25];
int cpimg[25][25];
int his[256];
int color[16];
```

```
// 计算整数 c 的压缩值
int compress(int c){
    int min=255,min_i=0;
    for(int h=0;h<16;h++){
        int k=abs(c-color[h]);
        if(k<min){
            min=k;
            min_i=h; //下标为转换值
        }
    }
    return min_i;
}
int main(){
    int n;
    cin>>n;
    char line[50]={0};
    for(int i=0;i<n;i++){
        cin>>line;
        for(int j=0;j<strlen(line);j+=2){
            int c = trans(line[j])*16+trans(line[j+1]);
            image[i][j/2]=c;
            his[c]++;// 统计灰阶值出现的次数
        }
    }
    // 在 his 数组中找出最多的前 16 种灰阶值保存在数组 color 中
    for(int i=0;i<16;i++){
        int max=0;
        for(int j=0;j<256;j++){
            if(his[max]<his[j]){
                max=j;
            }
        }
        color[i]=max;
        his[max]=-1;// 标记为-1，该灰阶值已获取
    }
    // 对 image 数组中的元素进行压缩处理
    int m=strlen(line)/2;
    for(int i=0;i<n;i++){
        for(int j=0;j<m;j++){
            cpimg[i][j]=compress(image[i][j]);
        }
    }
```

```
    // 输出选取的 16 个灰阶
    for (int i=0;i<16;i++)
        cout<<itrans(color[i]/16)<<itrans(color[i]%16);
    cout<<endl;
    // 输出压缩后的图像
    for (int i=0;i<n;i++) {
        for (int j=0;j<m;j++)
            cout<<itrans(cpimg[i][j]);
        cout<<endl;
    }
    return 0;
}
```