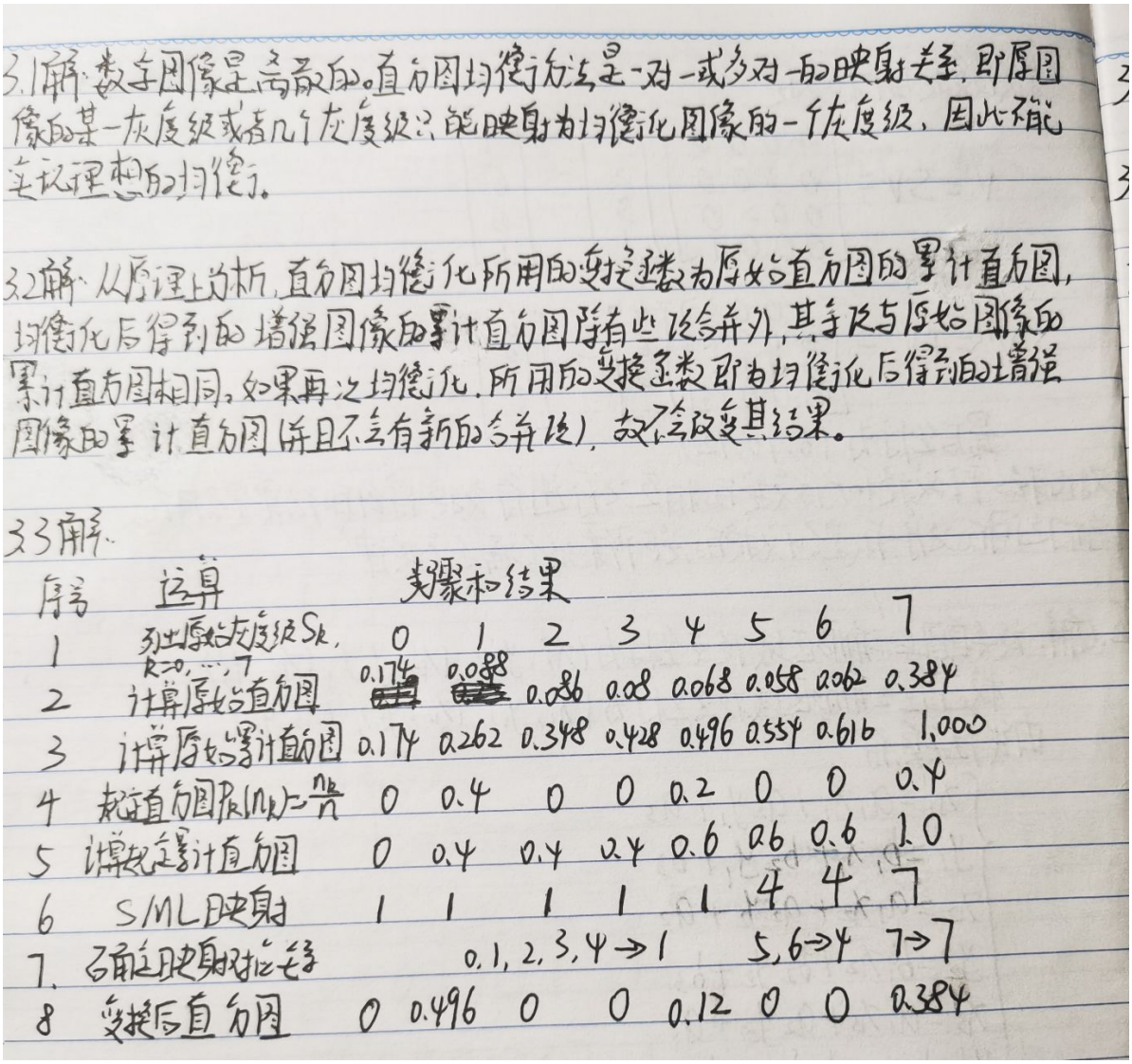
数字图像HW2

PB18000006 范翔宇

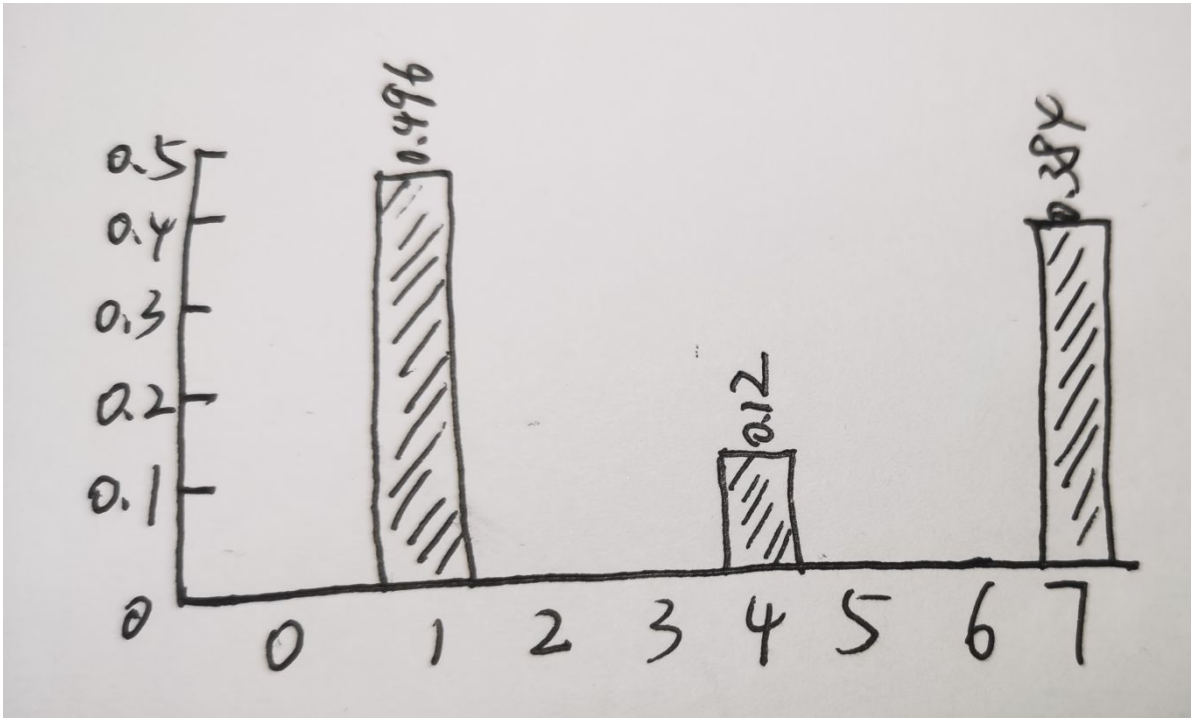


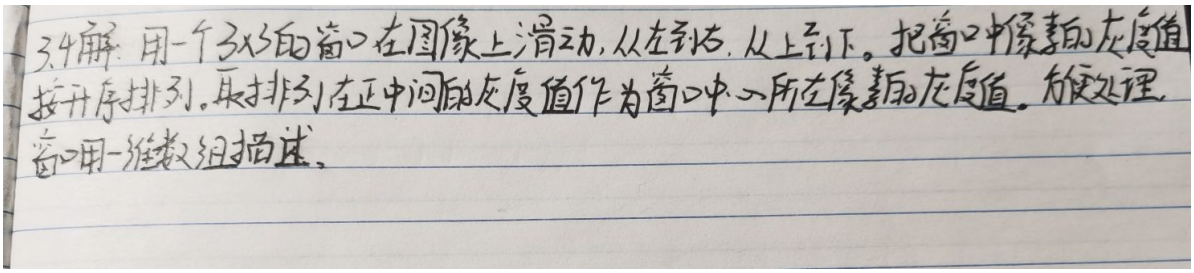
3.1解：数字图像是离散的。直方图均衡方法是一对一或多对一的映射关系，即原图像的某一灰度级或者几个灰度级只能映射为均衡化图像的一个灰度级，因此不能实现理想的均衡。

3.2解：从原理上分析，直方图均衡化所用的变换函数为原始直方图的累计直方图，均衡化后得到的增强图像的累计直方图除有些项合并外，其余项与原始图像的累计直方图相同。如果再次均衡化，所用的变换函数即为均衡化后得到的增强图像的累计直方图（并且不会有新的合并项），故不会改变其结果。

3.3解：

序号	运算	步骤和结果							
1	列出原始灰度级 S_k，$k=0,\cdots,7$	0	1	2	3	4	5	6	7
2	计算原始直方图	0.175	0.088	0.086	0.08	0.068	0.058	0.062	0.384
3	计算原始累计直方图	0.175	0.262	0.348	0.428	0.496	0.554	0.616	1.000
4	规定直方图及取值	0	0.4	0	0	0.2	0	0	0.4
5	计算规定累计直方图	0	0.4	0.4	0.4	0.6	0.6	0.6	1.0
6	SML映射	1	1	1	1	1	4	4	7
7.	合并映射相等项	0,1,2,3,4→1					5,6→4		7→7
8	变换后直方图	0	0.496	0	0	0.12	0	0	0.384

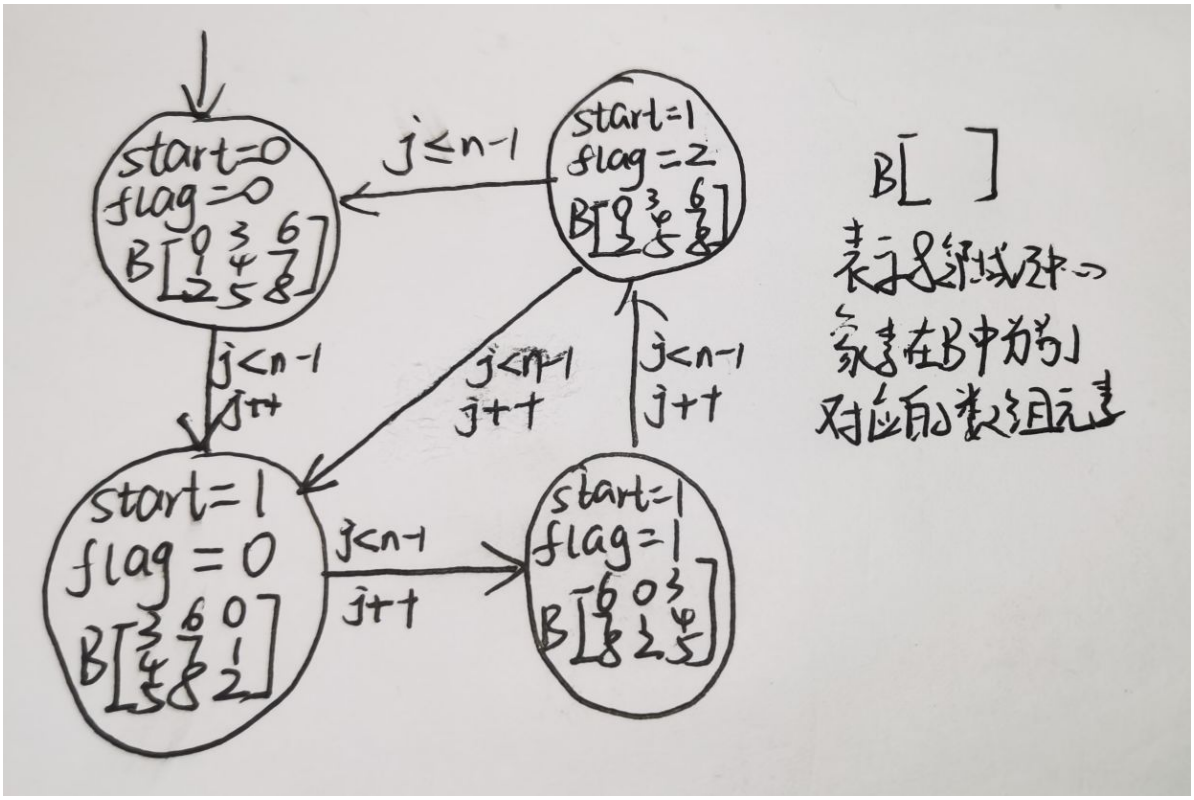




036036

147147

258258




```
//A[n][n]已知,下标为从0到n-1;B[10],C[n][n],D[10],flag,start全局变量也提前声明
```

```
void trans(int i, int j, int A[n][n], int B[10]){
    if(start == 0){
        B[0] = A[i - 1][j - 1];
        B[1] = A[i - 1][j];
        B[2] = A[i - 1][j + 1];
        B[3] = A[i][j - 1];
        B[4] = A[i][j];
        B[5] = A[i][j + 1];
        B[6] = A[i + 1][j - 1];
        B[7] = A[i + 1][j];
        B[8] = A[i + 1][j + 1];
        start = 1;
        flag = 0;
    }else if(flag == 0){
        B[0] = A[i + 1][j - 1];
        B[1] = A[i + 1][j];
        B[2] = A[i + 1][j + 1];
        flag = 1;
    }else if(flag == 1){
        B[3] = A[i + 1][j - 1];
        B[4] = A[i + 1][j];
        B[5] = A[i + 1][j + 1];
        flag = 2;
    }else{
        B[6] = A[i + 1][j - 1];
        B[7] = A[i + 1][j];
        B[8] = A[i + 1][j + 1];
        flag = 0;
    }
    return;
}
int main(){
    int i,j;
    //内部像素点
    for(i = 1;i < n - 1;i ++){
        start = 0;
        for(j = 1;j < n - 1;j ++){
            trans(i, j, A, B);
            copy(D,B);//将数组B的数据复制到D，再这里不细写具体实现
            qsort(D, 9, sizeof(int), compare);
            C[i][j] = D[4];
        }
    }
    //顶点处像素点
    C[0][0] = C[1][1];
    C[0][n - 1] = C[1][n - 2];
    C[n - 1][0] = C[n - 2][1];
    C[n - 1][n - 1] = C[n - 2][n - 2];
    //边缘非顶点处像素点
    for(j = 1;j < n - 1;j ++){
        C[0][j] = C[1][j];
        C[n - 1][j] = C[n - 2][j];
    }
    for(i = 1;i < n - 1;i ++){
        C[i][0] = C[i][1];
        C[i][n - 1] = C[i][n - 2];
    }
    copy(A,C);//将数组C的数据复制到A，再这里不细写具体实现
```

```
    return 0;
}
```