Assignment 2　　2152343 何慧琳

1. 顺序查找的平均键值比较次数：

$$C_{avg\_seq} = \sum_{i=1}^{10^6} i \cdot \frac{1}{10^6} \approx 5\times10^5$$

折半查找的平均键值比较次数：

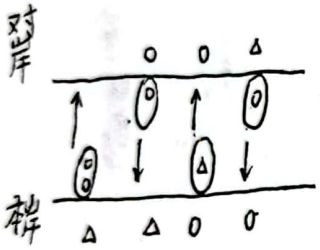
$$C_{avg\_bin} \approx \log_2 10^6$$

$$\frac{5\times10^5}{\log_2 10^6} \approx 2.51\times10^4$$

∴折半查找比顺序查找平均快 2.51×10^4 倍

2、经分析，每渡一名士兵，需要2个男孩先一起将船划到对岸留下一名男孩，另一个男孩将船划回给一名士兵渡河，士兵到岸后，留在对岸的男孩将船划回。

流程示意图如下：○为男孩，△为士兵



∴每渡一名士兵，船要在两岸间横渡4次

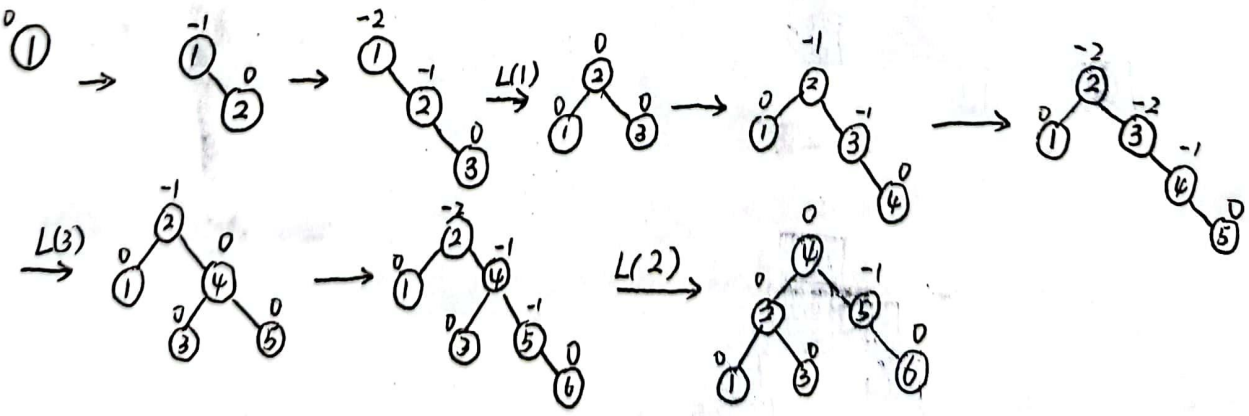
∴这条船要在岸与岸之间横渡 $4N$ 次

3. ⑧ 前序法：abdecf

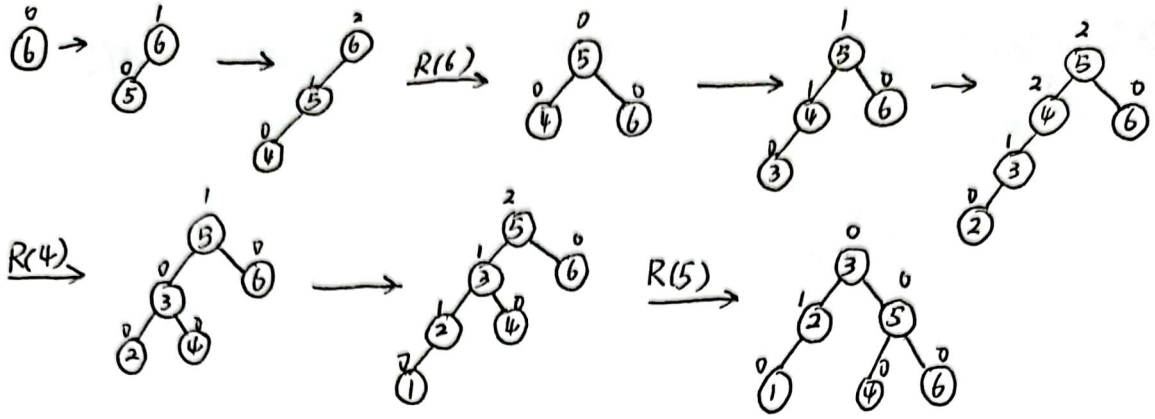
中序法：dbeacf

后序法：debfca

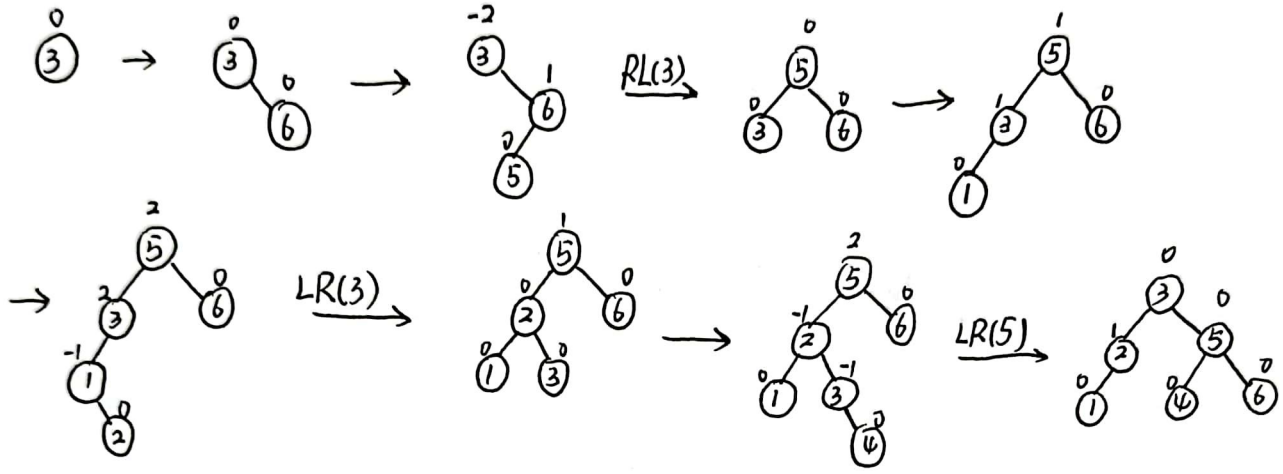
4. a.



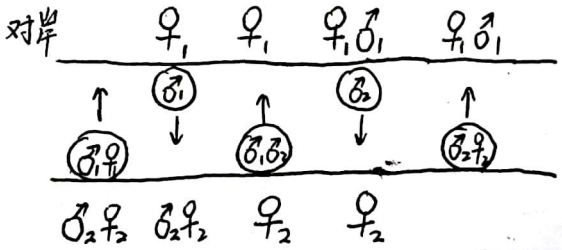
b.



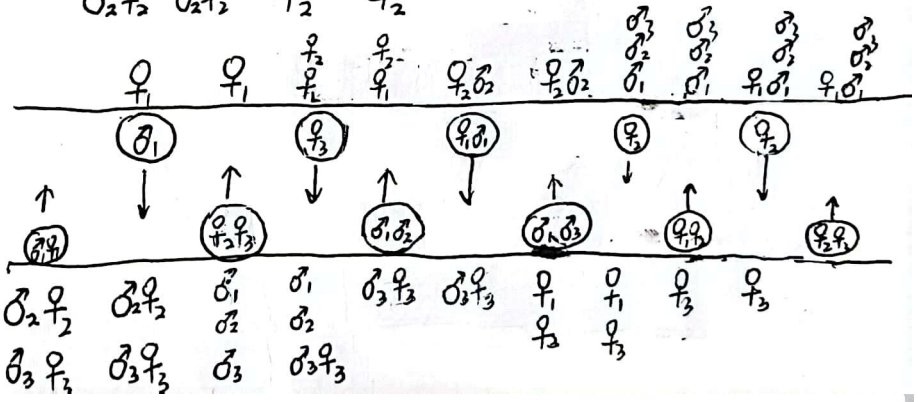
c.



5、a. 5次. 过程如下：



b. 11次



C. 无解. 由分析可知，在六次过河之后，有n-1次夫妇和船在本岸，一对夫妇在对岸. 在这种情况下只能再送一对夫妇过去从而回到第五次渡河后的状态.（因为如果送两妻子过去，对岸妻子数将大于丈夫数，不行；送两丈夫过去，由于n≥4，本岸将留下两妻子和一对夫妇，也不行）. 所以无法将全部夫妇送到对岸.

编程题

我尝试了两种方法：分治法（二分法）和减行/减列法。通过事先计算时间复杂度和编写代码测试实际耗时发现两种方法时间复杂度和耗时上都差不多，第一种方法由于采用了递归空间复杂度更高，所以最终采用第二种方法。

第一种方法：

1）思路：先选取m、n中较小的，以行数m较小为例，在待分区找到中间列对中间列进行二分找到最接近target的数并以此为分界点在左下和右上区域重复上述过程。

2）时间复杂度：O（min（m，n））

空间复杂度：与递归树的深度和矩阵大小有关

3）代码

```
bool if_find(vector<vector<int> >& matrix, int target,int left,int right,int top,int bottom,int row_or_col)
{
    int midi=(top+bottom)/2,midj = (right + left) / 2;
    if (left > right || top > bottom)
        return 0;
    if (row_or_col == 0) {//竖向二分
        int bottom1 = bottom, top1 = top;
        int bt;
        while ( bottom1- top1 >= 0) {
            midi = (top1 + bottom1) / 2;
            if (target == matrix[midi][midj])
                return 1;
            else if (target < matrix[midi][midj]) {
                bottom1 = midi-1;
                bt = 0;
            }
            else if (target > matrix[midi][midj]) {
                top1 = midi + 1;
                bt = 1;
            }
        }
        if (bt == 0) {
            if (if_find(matrix, target, left, midj-1, top1, bottom, 0))
                return 1;
            if (if_find(matrix, target, midj, right, top, bottom1, 0))
                return 1;
        }
        else {
            if (if_find(matrix, target, left, midj, top1, bottom, 0))
                return 1;
            if (if_find(matrix, target, midj + 1, right, top, bottom1, 0))
                return 1;
        }
```

续

```
    }
    else{//横向二分
        int left1 = left, right1 = right;
        int lr;
        while (right1 - left1>=0) {
            midj = (right1 + left1) / 2;
            if (target == matrix[midi][midj])
                return 1;
            else if (target < matrix[midi][midj]) {
                right1 = midj-1;
                lr = 1;
            }
            else if (target > matrix[midi][midj]) {
                left1 = midj + 1;
                lr = 0;
            }
        }
        if (lr == 0) {
            if (if_find(matrix, target, left, right1, midi+1, bottom, 1))
                return 1;
            if (if_find(matrix, target, left1, right, top, midi, 1))
                return 1;
        }
        else {
            if(if_find(matrix, target, left, right1, midi, bottom, 1))
                return 1;
            if(if_find(matrix, target, left1, right, top, midi-1, 1))
                return 1;

        }
    }
    return 0;
```

```
bool searchMatrix(vector<vector<int> >& matrix, int target){
        //TODO  二分法
    int m = matrix.size(),n=matrix[0].size();//获取行列数
    int left = 0, right = n-1,top=0,bottom=m-1,row_or_col=m<n?0:1;
    return if_find(matrix, target, left, right, top,bottom,row_or_col);
}
*/
```

4）运行结果

```
bool if_find(vector<vector<int> >& matrix, int target,int left,int right,int top,int bottom,int row_or_col) { ...
bool searchMatrix(vector<vector<int> >& matrix, int target){
        //TODO 二分法
    int m = matrix.size(),n=matrix[0].size();//获取行列数
    int left = 0, right = n-1,top=0,bottom=m-1,row_or_col=m<n?0:1;
    return if_find(matrix, target, left, right, top,bottom,row_or_col);
}
```

```
C:\WINDOWS\system32\cmd.exe
correct:200
error:0
用时:267ms
请按任意键继续. . .
```

```
bool if_find(vector<vector<int> >& matrix, int target,int left,int right,int top,int bottom,int row_or_col) { ... }
bool searchMatrix(vector<vector<int> >& matrix, int target){
        //TODO 二分法
    int m = matrix.size(),n=matrix[0].size();//获取行列数
    int left = 0, right = n-1,top=0,bottom=m-1,row_or_col=m<n?0:1;
    return if_find(matrix, target, left, right, top,bottom,row_or_col);
}
```

```
C:\WINDOWS\system32\cmd.exe
correct:200
error:0
用时:250ms
请按任意键继续. . .
```

第二种方法：

1）思路：从右上角元素开始比较，比target大则到右边的元素继续比较，比target小则到下面的元素继续比较，如此重复直到找到target或比较到越界元素。

2）时间复杂度：O（m+n）

空间复杂度：m*n

3）代码：

```
bool searchMatrix(vector<vector<int> >& matrix, int target) {
    //TODO 减列减行法
    int m = matrix.size(), n = matrix[0].size();//获取行列数
    int i = 0, j = n - 1;
    while (i < m && j >= 0) {
        if (target == matrix[i][j])
            return 1;
        else if (target < matrix[i][j])
            j--;
        else
            i++;
    }
    return 0;
}
```

4）运行结果：

```
bool searchMatrix(vector<vector<int> >& matrix, int target) {
    //TODO  减列减行法
    int m = matrix.size(), n = matrix[0].size();//获取行列数
    int i = 0, j = n - 1;
    while (i < m && j >= 0) {
        if (target == matrix[i][j])
            return 1;
        else if (target < matrix[i][j])
            j--;
        else
            i++;
    }
    return 0;
}

int main(){
```

```
C:\WINDOWS\system32\cmd.exe
correct:200
error:0
用时:260ms
请按任意键继续. . .
```

```
bool searchMatrix(vector<vector<int> >& matrix, int target) {
    //TODO  减列减行法
    int m = matrix.size(), n = matrix[0].size();//获取行列数
    int i = 0, j = n - 1;
    while (i < m && j >= 0) {
        if (target == matrix[i][j])
            return 1;
        else if (target < matrix[i][j])
            j--;
        else
            i++;
    }
    return 0;
}
```

```
C:\WINDOWS\system32\cmd.exe
correct:200
error:0
用时:266ms
请按任意键继续. . .
```