二分法实战练习精讲

主讲人 令狐冲

三种类型二分

在排序的输入集上二分

在未排序的输入集上二分

在答案集上二分

在排序输入集二分——排序矩阵找数问题

https://www.lintcode.com/problem/search-a-2d-matrix

一个矩阵每一行都有序，每一行都比前面一行所有数要大

在矩阵中寻找一个数是否存在

二维坐标转换一维坐标

每一行都比上一行大

因此矩阵从上到下从左到右列出来是有序的

n行m列的矩阵: **x, y → x * m + y**

套用二分法模板，使用子函数 get 进行坐标转换


```python
def searchMatrix(self, matrix, target):
    if not matrix or not matrix[0]:
        return False

    n, m = len(matrix), len(matrix[0])
    start, end = 0, n * m - 1
    while start + 1 < end:
        mid = (start + end) // 2
        if self.get(matrix, mid) < target:
            start = mid
        else:
            end = mid

    if self.get(matrix, start) == target:
        return True
    if self.get(matrix, end) == target:
        return True
    return False

def get(self, matrix, index):
    x = index // len(matrix[0])
    y = index % len(matrix[0])
    return matrix[x][y]
```

```java
public boolean searchMatrix(int[][] matrix, int target) {
    if (matrix == null || matrix.length == 0) {
        return false;
    }
    if (matrix[0] == null || matrix[0].length == 0) {
        return false;
    }

    int n = matrix.length, m = matrix[0].length;
    int start = 0, end = n * m - 1;
    while (start + 1 < end){
        int mid = start + (end - start) / 2;
        if (get(matrix, mid) < target) {
            start = mid;
        } else {
            end = mid;
        }
    }

    if (get(matrix, start) == target) {
        return true;
    }
    if (get(matrix, end) == target) {
        return true;
    }
    return false;
}

private int get(int[][] matrix, int index) {
    int x = index / matrix[0].length;
    int y = index % matrix[0].length;
    return matrix[x][y];
}
```

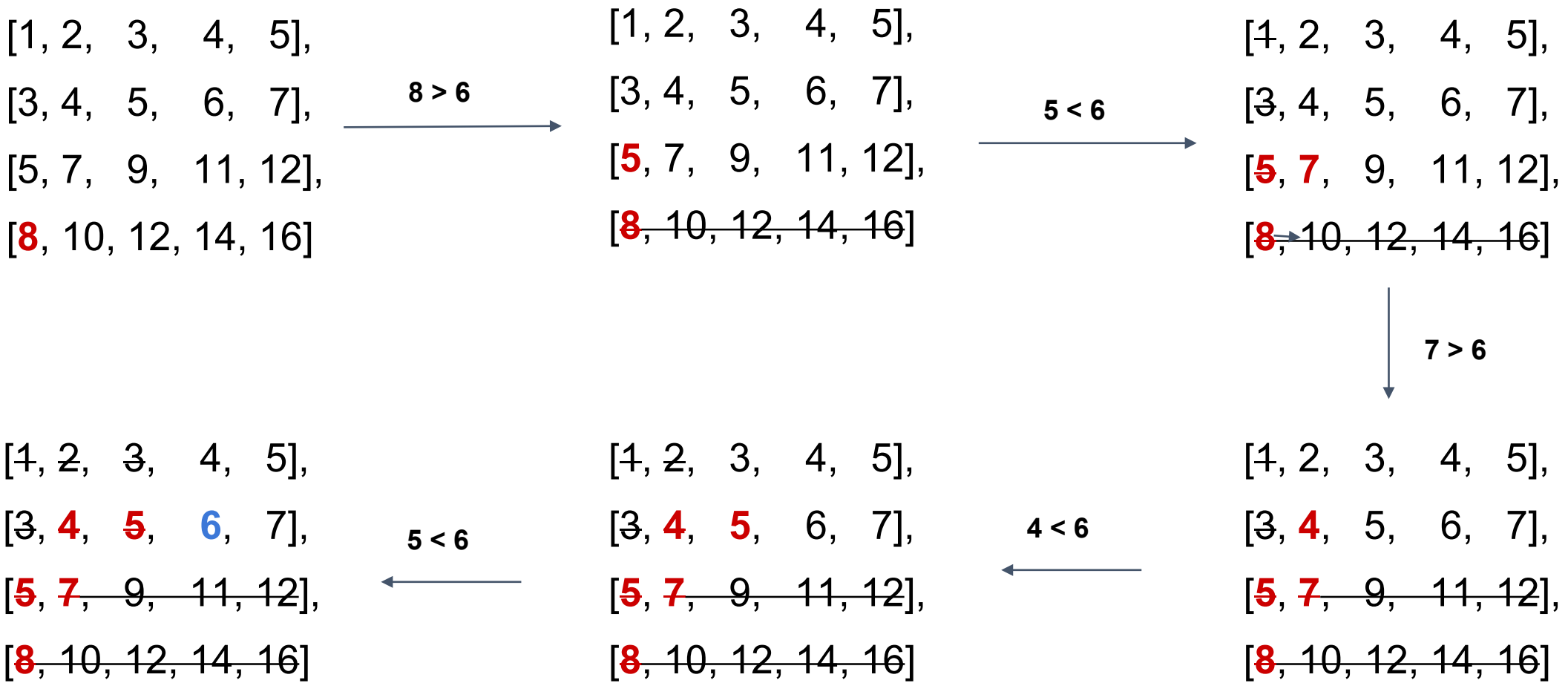
排序矩阵找数问题 II

https://www.lintcode.com/problem/search-a-2d-matrix-ii

每一行和每一列分别都是排好序的且**严格递增**

不保证下一行都比上一行的数大

从左下或者右上出发，找 6



记住就好


```python
def searchMatrix(self, matrix, target):
    if not matrix or not matrix[0]:
        return 0

    n, m = len(matrix), len(matrix[0])
    x, y = n - 1, 0
    count = 0
    while (x >= 0 and y < m):
        if matrix[x][y] == target:
            x -= 1
            y += 1
            count += 1
        elif matrix[x][y] > target:
            x -= 1
        else:
            y += 1

    return count
```

```java
public int searchMatrix(int[][] matrix, int target) {
    if (matrix == null || matrix.length == 0) {
        return 0;
    }
    if (matrix[0] == null || matrix[0].length == 0) {
        return 0;
    }

    int n = matrix.length;
    int m = matrix[0].length;
    int i = n - 1, j = 0, count = 0;
    while (i > -1 && j < m) {
        if (matrix[i][j] == target) {
            count++;
            i--;
            j++;
        } else if (matrix[i][j] < target) {
            j++;
        } else {
            i--;
        }
    }

    return count;
}
```

在未排序输入集二分——
包含全部黑色像素的最小矩阵

https://www.lintcode.com/problem/smallest-rectangle-enclosing-black-pixels/

矩阵中有黑色像素和白色像素分别用 1 和 0 表示

找到最小的矩阵，能把所有的 1 都框起来

假设所有的 1 是连通的

入口函数的要精简


```
def minArea(self, image, x, y):
    if not image or not image[0]:
        return 0

    n, m = len(image), len(image[0])
    left = self.find_first(image, 0, y, self.check_column)
    right = self.find_last(image, y, m - 1, self.check_column)
    up = self.find_first(image, 0, x, self.check_row)
    down = self.find_last(image, x, n - 1, self.check_row)

    return (right - left + 1) * (down - up + 1)
```

四个二分合并为两个二分


```python
def find_first(self, image, start, end, check_func):
    while start + 1 < end:
        mid = (start + end) // 2
        if check_func(image, mid):
            end = mid
        else:
            start = mid
    if check_func(image, start):
        return start
    return end

def find_last(self, image, start, end, check_func):
    while start + 1 < end:
        mid = (start + end) // 2
        if check_func(image, mid):
            start = mid
        else:
            end = mid
    if check_func(image, end):
        return end
    return start
```

```python
def check_column(self, image, col):
    for i in range(len(image)):
        if image[i][col] == '1':
            return True
    return False

def check_row(self, image, row):
    for j in range(len(image[0])):
        if image[row][j] == '1':
            return True
    return False
```

在答案集二分——抄书问题

https://www.lintcode.com/problem/copy-books/

k 个抄写员抄n本书，每个抄写员必须连续抄写挨着的几本书

问所有抄写员同时抄写，最少什么时候抄完

pages = [3, 2, 4], k = 2 可以在 5个时间单位内抄完

抄书问题


```java
public int copyBooks(int[] pages, int k) {
    if (pages == null || pages.length == 0) {
        return 0;
    }
    if (k == 0) {
        return -1;
    }

    int start = 0, end = Integer.MAX_VALUE;

    while (start + 1 < end) {
        int mid = start + (end - start) / 2;
        if (getNumberOfCopiers(pages, mid) <= k) {
            end = mid;
        } else {
            start = mid;
        }
    }

    if (getNumberOfCopiers(pages, start) <= k) {
        return start;
    }
    return end;
}
```

```java
private int getNumberOfCopiers(int[] pages, int limit) {
    int copiers = 0;
    int lastCopied = limit;

    for (int page : pages) {
        if (page > limit) {
            return Integer.MAX_VALUE;
        }
        if (lastCopied + page > limit) {
            copiers++;
            lastCopied = 0;
        }

        lastCopied += page;
    }

    return copiers;
}
```

其他答案

https://www.jiuzhang.com/solution/copy-books/

本题还可以使用动态规划实现