

1)

该算法能处理 3 点共线的问题, 预设参数, 1 个队列, 2 个栈, 但是效率可能会受影响。

拆分与执行:

1 判定: 如果集合 $S(2n)$ 中只有两个元素, 并且颜色不一样返回成功。

2 执行:

对所有点的集合 $S(2n)$ 以 Y 为基准做排序, 选取一个 Y 坐标最小的点, 如果有 Y 相同并列的坐标, 则选取 X 值在中间的点。记该点为 X_0 , 其余 $S(2n - X_0)$ 的点分别与 X_0 连线, 然后计算每个点对应的极角, 并以极角的大小为基准重新排序, 极角相同值的则按距离 X_0 的远近排序。设定一个变量 $tmp = 0$, $sum = 0$, 对新拍好序的集合 $(2n - X_0)$ 进行出队操作, 每个出队的点进行一次判断然后 $sum++$, 如果和 X_0 的颜色不一样则 $tmp + 1$, 否则若和 X_0 颜色相同则 $tmp - 1$, 然后将该点送进集合 S_1 , 进栈。如果 $tmp = -1$, 则记录下这个点并让其也进入匹配点栈 S_2 。

在所有点匹配完后对匹配点栈进行出栈操作, 设栈顶元素为 Y_0 , 然后将 S_1 中在 Y_0 上面的元素全部出栈, 进入 S 集合的队中, 再对 S_1 中的次栈顶元素和栈顶元素的极角做判断(就是 Y_0 和 Y_0 下面那个元素), 如果相等, 表示三点共线了, 该 Y_0 点不能要, 匹配点栈继续出栈操作找新的点作为 Y_0 , 重复刚才的操作, 如果不相等, 则表示 Y_0 点可以和 X_0 点连成线段, S_1 栈顶 y_0 出栈。对剩余的 S 集合和 S_1 集合进行迭代。

如果匹配栈已经空了, 但依然没有找到能和 X_0 匹配的 Y_0 点, 则返回失败, 不再进行迭代。

合并: 合并两个部分的返回结果, 如果有一个失败则返回失败, 两个都是成功才返回成功。

该算法不能处理 3 点共线的问题, 预设参数, 1 个队列, 1 个栈, 但是效率高。

1 判定: 如果集合 $S(2n)$ 中只有两个元素, 两个点坐标。

2 执行:

对所有点的集合 $S(2n)$ 以 Y 为基准做排序, 选取一个 Y 坐标最小的点, 如果有 Y 相同并列的坐标, 则选取 X 值在中间的点。记该点为 X_0 , 其余 $S(2n - X_0)$ 的点分别与 X_0 连线, 然后计算每个点对应的极角, 并以极角的大小为基准重新排序, 极角相同值的则按距离 X_0 的远近排序。设定一个变量 $tmp = 0$, $sum = 0$, 对新拍好序的集合 $(2n - X_0)$ 进行出队操作, 每个出队的点进行一次判断然后 $sum++$, 如果和 X_0 的颜色不一样则 $tmp + 1$, 否则若和 X_0 颜色相同则 $tmp - 1$, 然后将该点送进集合 S_1 , 进栈。如果 $tmp = -1$, 则记录下这个点, 如果 $sum > n$ 并且 tmp 不为空则进行下一步。

在匹配完后将最后一次记录的点记为 Y_0 , 然后将 S_1 中在 Y_0 上面的元素全部出栈, 进入 S 集合的队中, Y_0 点可以和 X_0 点连成线段, S_1 栈顶 y_0 出栈。对剩余的 S 集合和 S_1 集合进行迭代。

合并: 合并两个部分的黑白点对。

时间复杂度分析:

执行阶段 $O(n \lg n)$

递归式 $T(n) = 2T(n/2) + o(\lg n)$

master 定理不能用，只能递归展开.

$T(n) = n * (\lg n + \lg n/2 + \lg n/4 + \lg n/8 \dots \lg 1)$ 共 $\log_2 n + 1$ 个

$= n * (\log_2 n * \lg n - \lg 1 - \lg 2 - \lg 4 - \lg 8 \dots \lg n)$

$= n * (\lg^2 n - (1/2)\log^2 n)$

$= n * (\lg n)^2$

最终得出 $T(n) = o(n * (\lg n)^2)$

2)

拆分与执行：

1) 判定，如果集合中只有一个元素，该元素大于 0 则返回原数，否则返回 0

2) 执行，将集合 $S(n)$ 以中间点分成 2 个部分，每部分 $n/2 - 1$ 个元素，直接迭代两个部分。

合并：

记录左右集合返回的值，从中间点开始，往左延伸直到边界，依次累加求和，记录下累加和中的最大值，再从中点往右延伸直到边界，依次累加求和，记录下累加和中最大值。这两个最大值相加的和就是最大序列在中间的情况。

最后返回三个值（左集合返回值，右集合返回值，与中间最大值）中最大的那个。

3)

拆分执行：

1) 判定：如果集合中的元素为 1 个而且这个数是 0 则返回 0，其余全部返回 1（因为 0 无法单独解码）

2) 执行：对于集合 $S(n)$ 取中间点，对其进行判断，如果这个数是 0,3,4,5,6,7,8,9 或者这个数是 2 但是右边的下一个数是 7,8,9,则表示这个数无法和后面的数组组合，只有一种表示法。将集合拆成 $(1 \rightarrow n/2)$ 和 $(n/2+1 \rightarrow n)$ 这两个集合，并将其标记成情况 1，迭代两个部分。

对于剩下的情况，即中间的数可以和后面一个数组成 2 位数，则情况有所变化，多了一种考虑的情况，将其分成 2 种情况，4 个集合。 $(1 \rightarrow n/2)$ 和 $(n/2+1 \rightarrow n)$ 是一种情况 $(1 \rightarrow n/2 - 1)$ 和 $(n/2 + 2 \rightarrow n)$ 是一种情况。将其记为情况二，然后分别对四个集合迭代递归。

合并：

对于情况一返回 $(1 \rightarrow n/2) * (n/2 + 1 \rightarrow n)$

对于情况二返回 $(1 \rightarrow n/2) * (n/2 + 1 \rightarrow n) + (1 \rightarrow n/2 - 1) * (n/2 + 2 \rightarrow n)$

4)

拆分与执行：

判定：

如果集合中只有 1 个元素则返回 0；

拆分执行

将集合分成($1 \rightarrow n/2$)与($n/2+1 \rightarrow n$)两个部分，迭代执行。

合并：

记录左右两个部分的返回结果记为 A,B。并定义变量 $C = 0$

将左右两个集合各自排序，每次取出两个集合中最小的元素。

如果取出的是左集合的元素则不做任何操作，如果取出的是右集合的元素，则 C 变量加上左集合元素中剩余元素的个数。

最后返回 $A + B + C$ 的结果。

5)

思路：友谊对就是指任意两点其间的矩形框里不含任何一个点，我们所要做的就是不停地拆分区域，直到找到这样的矩形框，使其间不存在点。

拆分与执行：

判定：如果该范围内没有任何一个点，并且该范围不含边界(总的边界不在这个范围内)则返回 1，否则返回 0；

拆分执行：先接受一个边界，然后接受一个集合 S 并进行以 X 坐标为基准的排序，找到 X 坐标中间的那个点，以其为基准沿 X 轴，Y 轴方向划线。将平面区分成 4 个象限，记作，1,2,3,4 象限。然后由这 4 个象限可以组成 4 个区域，12 共同构成上区域，23 共同构成左区域，34 共同构成下区域，14 共同构成右区域，对各个区域内的点进行统计，各自构成新的集合，然后以各自的区域作为新的边界迭代递归下去。

合并：返回 上区域返回结果 + 左区域返回结果 + 下区域返回结果 + 右区域返回结果。