前缀和 · 差分 · 离散化 算法笔记

> 这三种思想在算法题中经常**组合使用**，尤其是在区间修改、区间统计、坐标范围极大的问题中（如矩形面积并、扫描线等）。

一、前缀和（Prefix Sum）

1. 核心思想

前缀和是一种 **用空间换时间** 的思想，通过预处理数组，使得任意区间的和可以在 **O(1)** 时间内得到。

---

2. 一维前缀和

定义

给定数组 `a[1..n]`，定义前缀和数组 `s`：

```
s[i] = a[1] + a[2] + ... + a[i]
```

区间查询

区间 `[l, r]` 的和：

```
s[r] - s[l-1]
```

---

3. 二维前缀和

常用于矩阵区域求和。

定义

```
s[i][j] = a[1..i][1..j] 的总和
```

计算公式

```
s[i][j] = s[i-1][j] + s[i][j-1] - s[i-1][j-1] + a[i][j]
```

查询子矩阵 `(x1,y1)` 到 `(x2,y2)`

```
s[x2][y2]
- s[x1-1][y2]
- s[x2][y1-1]
+ s[x1-1][y1-1]
```

---

4. 适用场景

- 多次区间求和
- 图像 / 矩阵区域统计
- 与差分结合使用

二、差分（Difference Array）

1. 核心思想

差分是 **前缀和的逆操作**，用于高效处理"**对一整段区间进行修改**"的问题。

---

2. 一维差分

定义

给定原数组 `a`，构造差分数组 `d`：

```
d[i] = a[i] - a[i-1]
```

区间加法

对区间 `[l, r]` 加上 `k`：

```
d[l] += k
d[r+1] -= k
```

最终数组可通过对 `d` 求前缀和还原。

---

3. 二维差分

常用于 **矩形区域整体加减。**

对矩形 `(x1,y1)` 到 `(x2,y2)` 加 `1`

```

```

```
d[x1][y1]     += 1
d[x2][y1]     -= 1
d[x1][y2]     -= 1
d[x2][y2]     += 1
```

再对 `d` 做二维前缀和即可得到覆盖次数。

---

4. 适用场景

- 区间更新 + 最终统计
- 矩形覆盖、涂色问题
- 与离散化结合处理大坐标

三、离散化（Discretization）

1. 为什么需要离散化?

当数据范围很大（如 `1e8`、`1e9`），但**实际使用的坐标数量很少**时：

- 直接开数组会 **爆内存**
- 实际信息是"稀疏的"

👉 离散化的目标：

> **只关心"出现过的位置"**

---

2. 基本步骤

1. 收集所有可能用到的坐标
2. 排序
3. 去重
4. 用 `lower_bound` 映射为下标

---

3. 示例（坐标离散化）

原坐标：

```
[0, 2, 4, 6]
```

离散后：

```
0 -> 0
2 -> 1
```

```
4 -> 2
6 -> 3
```

---

4. 面积类问题的关键点 ⚠

离散化 **不是等距的**，所以：

- 一个格子 `(i,j)` 实际代表的面积是：

```
(x[i+1] - x[i]) * (y[j+1] - y[j])
```

👉 **不能直接把 1 当作单位面积！**

四、三者结合的经典模型

矩形面积并 / 覆盖问题

常见套路：

1. 对 `x`、`y` 坐标离散化
2. 用二维差分标记矩形覆盖
3. 用二维前缀和还原覆盖次数
4. 统计 `cover > 0` 的格子面积

五、总结对比

技术	解决问题	优点
前缀和	快速区间查询	查询 O(1)
差分	快速区间修改	修改 O(1)
离散化	大范围小数据	降低空间复杂度

📌 **一句话记忆**：

> 修改用差分，查询用前缀，坐标太大先离散。