4. 前缀和与差分

张桄玮(gwzhang@cug.edu.cn)

郑州一中(Legacy)

2024-08-01

从运算说起

运算 = 函数

- $a+b$实际上是我们有一个二元函数$\mathrm{add}(a,b)$
  - $\mathrm{add}:\mathbb{R}\times\mathbb{R}\to\mathbb{R}$
- 就像 C++大家写的 `function` 一样
- 许多时候的问题: 将一个运算反复多次

```
add(add(add(add(1,2),3),4),5)
```

问题: 这段表达式用加法怎么写?

- 两种记号的优缺点:
  - C++风格: 很轻松处理多元函数(多加几个逗号);
  - 日常的风格: 二元函数
    - 交换律, 结合律可以很好地被体现.

不断对一个列表相加 = 求和

- 对一个“不定长的东西”施以求和

```
// T 是某种类型, f 是某个函数
T sum(vector<T> list){
  T result = 零元;
  for(int i=0; i<list.length(); i++){
    result += f(list[i]);
  }
  return result;
}
```

数学上面的记号:

$$\sum_{i \in \text{list}} f(i)$$

Python: `sum(map(list, lambda x: ...))`

- 先把列表中的元素映射成想要的值, 然后加起来.

求和记号

定义 01: 对于一个可数的集合$S = \{a_1, a_2, ..., a_n\}$, 求和实际上是将这个集合的每一个元素在某一个函数$f : S \to X$作用之后, 把对应的值相加. 记作$\sum_{i \in S} f(i)$; 表示$f(a_1) + f(a_2) + f(a_3) + ... + f(a_n) + ...$

感到恐慌的时候核心指导原则

- 把求和记号写成$\sum_{i \in \text{ list}} f(i)$ 的形式
- 把求和记号拆开成普通的连加号

接下来: 考察一些求和记号的性质

a) 换元法(变量代换)

- 如果将 $\sum_{i \in S} f(i)$ 换为 $\sum_{k \in S} f(k)$, 求和的表示内容将不变.

求和记号

- "当前 S 中的代表" 用来表示的字母不同, 从而肯定不会影响整个映射 f 所表达的意思.

例子:

$$\sum_{1\le k\le n} a_k \xlongequal{k\ \text{代换为}s+1} \sum_{1\le s+1\le n} a_{s+1} \xlongequal{s\ \text{代换为}k} \sum_{1\le k+1\le n} a_{k+1}$$

- 经常对于整数的指标求和, 所以 $\sum_{a\le i\le b} f(i)$ 也写作 $\sum_{i=a}^{b} f(i)$.

这样在做变量代换的时候就会发生

$$\sum_{i=1}^{n} a_i \xlongequal{i\ \text{代为}i+1} \sum_{i=0}^{n-1} a_{i+1}$$

更易于出错."换元必换上下限(这绝不是你最后一次遇到这个)"

相当基本: 做抽象, 做简化

求和记号

变量的代换

- 记号: 用$k := k+1$来表示把k代为$k+1$.
- 条件: “自由” 的变量(没有具体地特指某一件事儿)

例子: 初中数学: “上加下减, 左加右减” 感觉不对称?

- 变量代换

例子: 说说下列的推导为什么错了?

$$\left(\sum_{j=1}^{n} a_j\right)\left(\sum_{k=1}^{n} \frac{1}{a_k}\right) = \sum_{j=1}^{n}\sum_{k=1}^{n} \frac{a_j}{a_k} = \sum_{k=1}^{n}\sum_{k=1}^{n} \frac{a_k}{a_k} = \sum_{k=1}^{n} n = n^2$$

.

b) 记号带来的性质

求和记号

性质 *0.1.*: 设K是某一有限个正整数的集合, 我们有如下的三条规则:

- 常数项进出求和记号:

$$\sum_{k\in k} cf(k) = c\sum_{k\in k} f(k);$$

- 求和记号的拆分:

$$\sum_{k\in K} f(k) + g(k) = \sum_{k\in K} f(k) + \sum_{k\in K} g(k);$$

- 若$p(k)$应用于K中的每一个元素之后组成的集合依然是K的一个排列, 那么

求和记号

$$\sum_{k\in K} f(k) = \sum_{p(k)\in K} f(p(k)).$$

例子：如$K=\{-1,0,1\}, p(k)=-k$, 由于 $p(-1)=1, p(0)=0$,$p(1)=-1$, p 对 k 中每一个元素构成集合为 $\{-1,0,1\}=K$.

- 老师讲的: “等差数列求和” 的故事, 求和记号的三条性质可以严肃地捕捉

例子 (等差数列求和):

求

$$S=\sum_{0\le k\le n}(a+bk)$$

的值.

考虑

$$
\begin{aligned}
S_2 &= \sum_{0\le k\le n}(a+bk) \\
&\xlongequal{k:=n-k} \sum_{0\le n-k\le n}(a+b(n-k)) = \sum_{0\le k\le n}(a+bn-bk)
\end{aligned}
$$

令$S+S_2$ 得

$$
\begin{aligned}
S+S_2=2S &= \sum_{0\le k\le n}(a+bk) + \sum_{0\le k\le n}(a+bn-bk) \\
&= \sum_{0\le k\le n}(2a+bn) = (2a+bn)\sum_{0\le k\le n}1 = (2a+bn)(n+1)
\end{aligned}
$$

那

求和记号

$$S = \frac{(n+1)(2a+bn)}{2}.$$

求区间和

给定一个数列a, 希望求出$\sum_{i=l}^{r} a_i$?

- 一个 for 循环即可
- 如果有10^5的询问?

定义前缀和数组s, 满足$s_i = \sum_{i=1}^{i} a_i$.

- 要得知$[l..r]$的和, 只需要$s_r - s_{l-1}$
  - 为什么? 试着推导一下

问题: 能不能仿照这前缀和, 发明一个前缀 "积" 出来?

关键: 对列表一部分施以**加法**, 任何需要的区间可以从一个较大的区间**减去**得到我想要的值.

对应: 加法运算意义下 0 相当于乘法中的谁? 加法的逆运算是减法, 乘法的逆运算是谁?

求区间和

- 一个数 +0 就像是 一个数 $\times 1$.
- 乘法的逆运算是除法

思路: 维护$s_i := a_1 a_2 ... a_i$, 要求出某一个区间的时候用除法.

如果对于一个列表做了映射之后的操作是乘法, 就可以写作

$$\prod_{i \in \text{ list}} f(i)$$

这就是为什么没有另外花费篇幅介绍$\prod$的各种性质

- 取一下log就又变成求和了...!

格局打开: 奇形怪状的运算

例子：定义正方形纸片上的运算$\curvearrowright$表示把正方形纸片顺时针旋转 90 度.

- 逆运算:
  - 把正方形纸片逆时针旋转 90 度
  - $\curvearrowright^{-1} = \curvearrowright\curvearrowright\curvearrowright$

问题：现在有编号为 $0 \sim 10$ 一共 10 个球, 我们现在有若干个区间的对换. 具体地, 对于区间 $[l..r]$ 的对换之后, 如果原来这方面的球的编号是 $\cdots, a_l, a_{l+1}, \cdots, a_r, \cdots$, 那么经过这次对换之后，这个区间的球的顺序就变成了 $\cdots, a_{l+1}, a_{l+2}, \cdots, a_r, a_l, \cdots$ 。现在你有 n 条操作规则, 每条操作规则就是两个数 l, r. 现在, 我们想知道你连续执行编号 a 到编号 b 的操作规则之后，得到的内容是多少. 注意

格局打开: 奇形怪状的运算

有 m 次查询. 数据范围: $1 \le n, m \le 10^5, 0 \le a, b \le 9, 1 \le l \le r \le n$.

- 数: 交换的左端点和右端点
- 运算: 就是我们的交换操作
- 运算的逆运算: 交换, 然后再交换

JOI2007A 最大的和

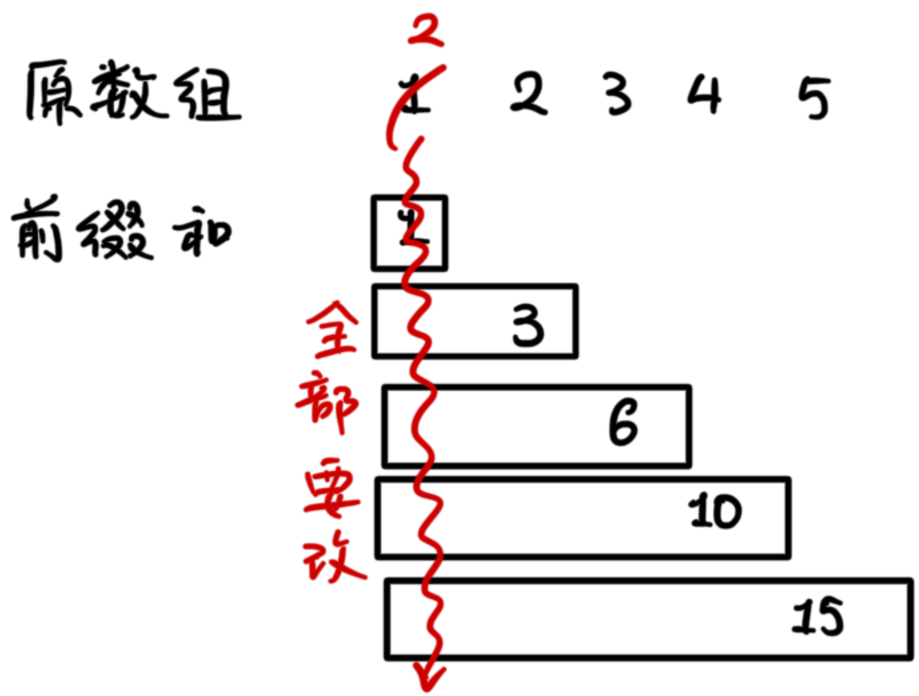
问题：读入 n 个整数的数列 $a_1, a_2, ..., a_n$ 和正整数 $k(1 \le k \le n)$, 请输出连续排列的 k 个整数的和的最大值。

- 维护前缀和
- 然后只要$O(n)$的时间得到想要的值!

回顾: 前缀和

对什么操作比较好? 对什么操作支持得不好?

- 这个数列最好是固定下来的, 不要有变动产生!



- 对于区间查询支持的很好!

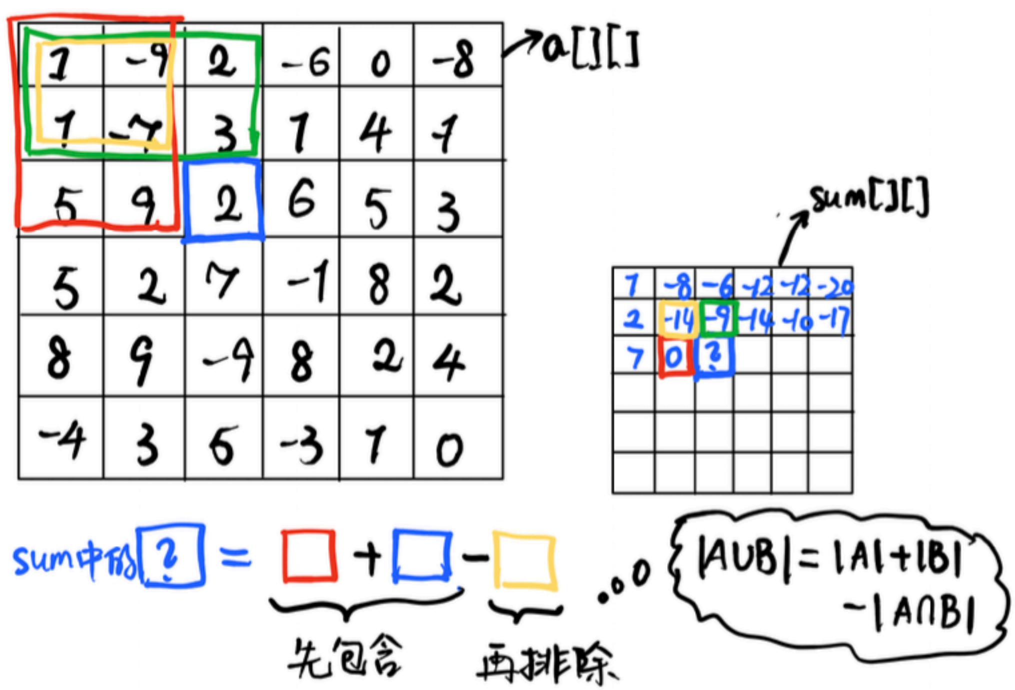
前缀和

二维前缀和

刚刚介绍了一维前缀和.

从一维前缀和到二维前缀和.

- 考虑$S[i][j] :=$ 第i行第j列左上方所有元素的和



实际上这是容斥原理$n = 2$的情况.

二维前缀和


```
void prefixSum2D(int a[][C]){
  int s[R][C];
  s[0][0] = a[0][0];

  for (int i = 1; i < C; i++)
    s[0][i] = s[0][i - 1] + a[0][i];
  for (int i = 1; i < R; i++)
    s[i][0] = s[i - 1][0] + a[i][0];

  for (int i = 1; i < R; i++) {
    for (int j = 1; j < C; j++)
      s[i][j] = s[i - 1][j] + s[i][j - 1]
                            - s[i - 1][j - 1] + a[i][j];
  }

}
```

- 还是一种递归的情况

P1387 最大正方形

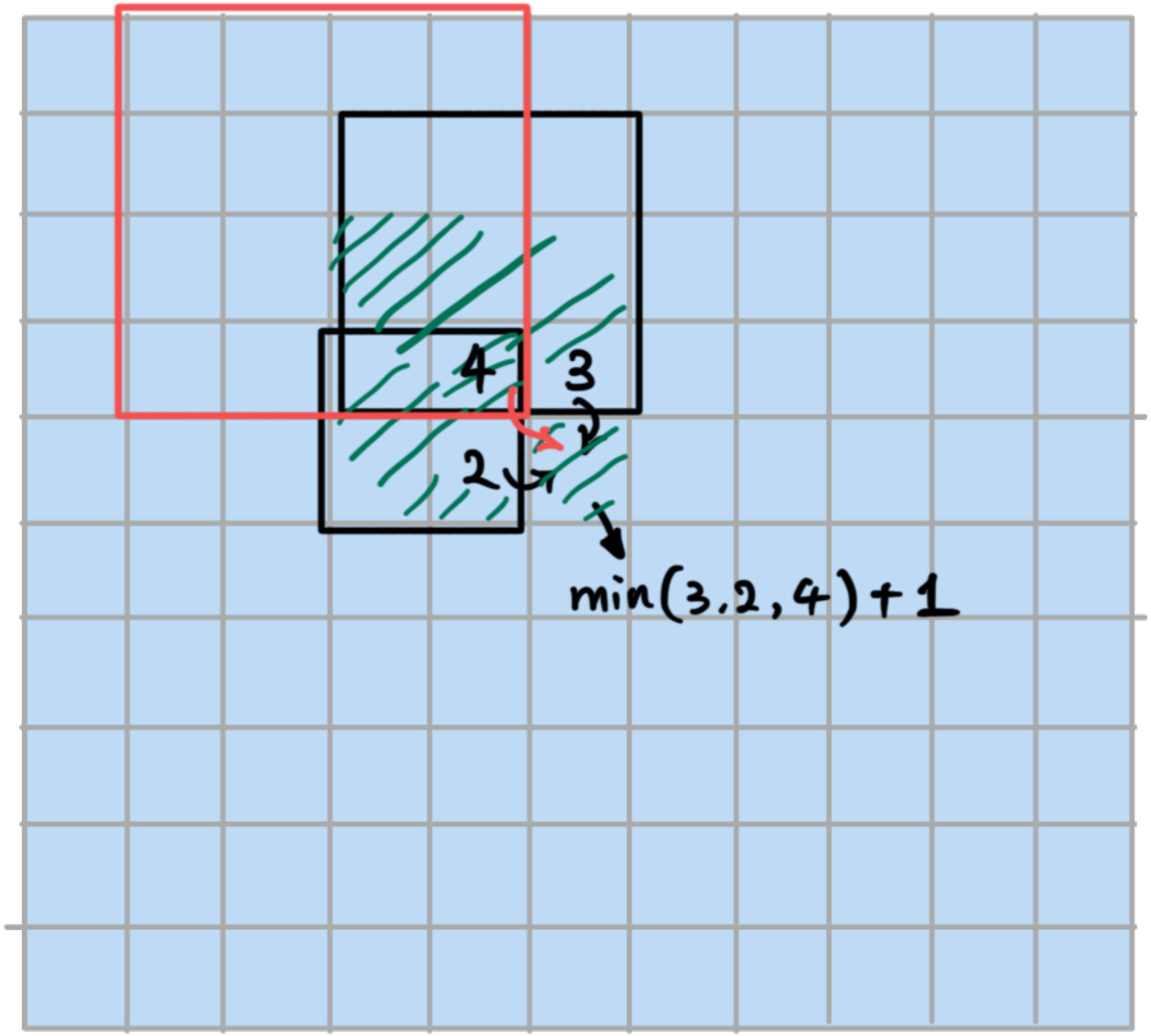
问题：在一个 $n \times m$ 的只包含 0 和 1 的矩阵里找出一个不包含 0 的最大正方形，输出边长。

- 想法: 把从每一个最小的正方形开始 "扩张"
- 注意扩张的时候能不能复用前面的内容?

要想从上往下扩展，上面的是扩展好的，新发现的节点一定靠右下角，然后看是不是能够合并

- 定义 $f[i][j] :=$ 以 (i, j) 为正方形的右下角，可以构成正方形的边长的最大值.
  - Base case: 只有 $a[i][j] = 1$ 的时候才能够作为正方形的右下角节点
  - Inductive case: 考虑可以从哪里扩展?

P1387 最大正方形



$$f[i][j] = \min(f[i][j-1], f[i-1][j], f[i-1][j-1]) + 1, \text{若} a[i][j] = 1$$

P1387 最大正方形

- 回顾刚才: 两个重要的因素
  - 状态的定义(如何定义子过程?)
  - 状态的转移
    - Base case 是什么?
    - 转移的顺序?

P2280 激光炸弹

问题：一种新型的激光炸弹, 可以摧毁一个边长为 m 的正方形内的所有目标。现在地图上有 n 个目标, 用整数 x_i, y_i 表示目标在地图上的位置, 每个目标都有一个价值 v_i 。激光炸弹的投放是通过卫星定位的, 但其有一个缺点, 就是其爆破范围, 即那个边长为 m 的边必须与 x 轴, y 轴平行。若目标位于爆破正方形的边上, 该目标不会被摧毁。

- 问最多可以摧毁多少目标?

- 枚举所有可能的正方形, 看看哪个最大?
- 枚举的时候要使用前缀和

P1083 借教室

问题：需要处理接下来 n 天的借教室信息，其中第 i 天学校有 r_i 个教室可供租借。共有 m 份订单，每份订单说的是某租借者需要从第 s_j 天到第 t_j 天租借教室（包括第 s_j 天和第 t_j 天），每天需要租借 d_j 个教室。

先到先得，也就是说我们要按照订单的先后顺序依次为每份订单分配教室。如果在分配的过程中遇到一份订单无法完全满足，则需要停止教室的分配，通知当前申请人修改订单。这里的无法满足指从第 s_j 天到第 t_j 天中有至少一天剩余的教室数量不足 d_j 个。

现在我们需要知道，是否会有订单无法完全满足。如果有，需要通知哪一个申请人修改订单。

P1083 借教室

朴素思路:

- 每个订单就是一个区间操作: 把开始时间~结束时间的教室数量减去借的教室数量.

问题: 前缀和适合不修改, 区间查询

当前的问题: 区间修改, 单点查询.

- 把一个区间修改之后什么改变的少一些?

$$\begin{array}{rccccc}
\text{原数组} & 2 & [3 & 4 & 6] & 8 \\
\text{差分数组} & \sim & 1 & 1 & 2 & 3 \\
+2\ \text{的数组} & 2 & [5 & 6 & 8] & 8 \\
+2\ \text{之后的差分数组} & \sim & 3 & 1 & 2 & 0
\end{array}$$

- 区间里面的后一个数比前一个数差的不会变

P1083 借教室

如何在差分数组上面表示原数组上的“区间加”?

- $[l..r]$上加x
- $d[l] \leftarrow d[l] + x; d[r+1] \leftarrow d[r+1] - x;$

如何得到原数组?

- 对差分数组求前缀和
- 需要一个初始条件a_1.

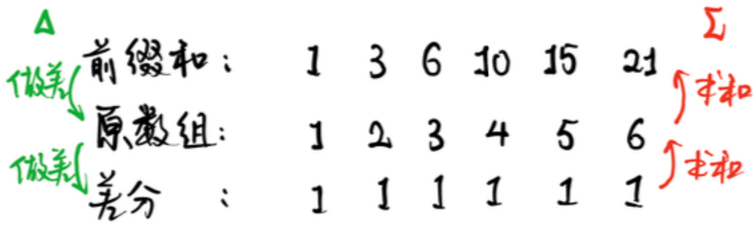
但是这道题还不够

- 二分答案: 答案满足单调性

差分

差分和前缀和互为逆运算

- 前缀和 = 随着序列的累积过程
- 差分 = 序列的变化趋势



> 问题：要在区间加一个等差数列怎么做？

差分一次不够，差分两次如何？

差分和前缀和互为逆运算

$$
\begin{aligned}
\text{原数组} &\ 1,2,3,\quad 4,5,6 \\
\text{差分一次} &\ 1,1,1,\quad 1,1,1 \\
\text{差分两次} &\ 1,0,0,\quad 0,0,0 \\
\text{在}[1..3]\text{加上等差数列}(2,4,6) &\ 3,6,9,\quad 4,5,6 \\
\text{差分一次} &\ 3,3,3,-5,1,1 \\
\text{差分两次} &\ 3,0,0,-8,6,0
\end{aligned}
$$

- 等于说对差分 1 次的数组又做区间加
- 那就再次差分
  - 记得二次差分数组在修改的地方有 2 处!

(可能还需要更好的数据结构维护两次前缀和, P1438 无聊的数列)

如何获得一般规律? 数学与符号来帮我们啦!

差分和前缀和互为逆运算

问题：要在区间加一个数列的平方怎么做?

差分 3 次!

- 根据刚刚的推导，你能说为什么差分 3 次就可以化为区间加的问题了吗?

差分的记号

定义 02：对于一个数列, 定义其差分算子$\Delta f(x)$为

$$\Delta f(x) := f(x+1) - f(x).$$

- 算子作用在函数上, 给出新的函数. 其本质就是从函数到函数的映射.

(a) 某些多项式的差分

1. 下降幂

如果我们求

$$\Delta(\underbrace{x(x-1)\cdots(x-m+1)}_{m\text{项}})$$

就会得到

差分的记号

$$\Delta(\underbrace{(x(x-1)\cdots(x-m+1))}_{m\text{项}}$$

$$= (x+1)x\cdots(x-m+2) - x(x-1)\cdots(x-m+1)$$

$$= m\underbrace{(x-1)\cdots(x-m+2)}_{m-1\text{项}}$$

说明Δ算子在上述的多项式上面作用有与求导算子在x^m次多项式有类似的效果.

定义 03：我们先定义

$$\underbrace{x(x-1)\cdots(x-m+1)}_{m\text{项}}$$

为x的m次下降幂. 记作$x^{\underline{m}}$

差分的记号

从而有

$$\Delta x^{\underline{m}} = m x^{\underline{m-1}}.$$

(b) 差分之后还是原来的函数的函数

希望找一个 $\Delta f(x) = f(x)$. 实际上,

$$f(x+1) - f(x) = f(x) \Leftrightarrow f(x+1) = 2f(x), \text{即}$$
$$f(x) = 2^x.$$

(c) 差分带来的运算

- 扔掉求和的上下标, 研究更广泛的一簇内容!

差分的记号

$$
\begin{array}{ll}
f = \sum g & \boxed{\Delta f} = g. \\
x^{\underline{m}} & m x^{\underline{m-1}} \\
2^x & 2^x \\
c^x & (c-1)c^x \\
c \cdot u & c \cdot \Delta u \\
u + v & \Delta u + \Delta v \\
uv & u\Delta v + Ev\Delta u
\end{array}
$$

例子 (分部求和法): 注意到

$$
\begin{aligned}
&\Delta(u(x)v(x)) = u(x+1)v(x+1) - u(x)v(x) \\
&= u(x+1)v(x+1) - u(x)v(x+1) \\
&+u(x)v(x+1) - u(x)v(x) \\
&= u(x)\Delta v(x) + v(x+1)\Delta u(x)
\end{aligned}
$$

若定义$Ef(x) := f(x+1)$, 那么

差分的记号

$$\Delta(uv) = u\Delta v + (Ev)\Delta u.$$

此时, 对两边同时取$\sum$, 有

$$\sum u\Delta v = uv - \sum(Ev)\Delta u.$$

二维差分

- 二维平面的方向可就多了! 凭什么选定某一个顺序

转换思路: 差分是前缀和的逆运算

- 差分的前缀和应该等于原数组

怎么办? 考虑一点一点构造差分矩阵

1. 初始状态: 原矩阵 A 是全 0 矩阵, 那其差分矩阵 B 显然也是全 0 矩阵
2. 归纳状态: 往原矩阵填入真实值, 顺便更新差分矩阵
   - 例如: 题目中原矩阵的第 $(2, 8)$ 个位置应该是 5 , 那相当于将原矩阵以 $(2, 8)$ 为左上角, 同样以 $(2, 8)$ 为右下角的所有元素加上了 5 。
   - 即: $B[2][8] \leftarrow B[2][8] + 5; B[3][8] \leftarrow B[3][8] - 5; B[2][9] \leftarrow B[2][9] - 5; B[3][9] \leftarrow B[3][9] + 5.$

格局打开: 微积分

如果不再是离散的数列($f : \mathbb{Z} \to \mathbb{R}$), 而是连续的函数($f : \mathbb{R} \to \mathbb{R}$)

- 就打开了微积分的大门(强烈推荐 3Blue1Brown 的可视化系列视频!)