

前缀和

一些情况下，对数组的子区间和进行多次询问，遍历是一个费时的行为，前缀和在这里就很有用。

多维前缀和



在一个二维数组中要求一个方形区域的和。这里同样可以运用前缀和。
 例如这张图，求绿色部分，就可以用整个减去横侧和纵侧的条，再加上被减两次的块。
 三维数组中同理，只是算式略不同。就是容斥定理的应用。
 但是随着维度t变高，计算前缀和的容斥的复杂度是 2^t 总复杂度达到了 $O(n^t \times 2^t)$

以三维为例：

```

for(int i=1;i<=n;i++)
  for(int j=1;j<=m;j++)
    for(int k=1;k<=p;k++)
      b[i][j][k]=b[i-1][j][k]+b[i][j-1][k]+b[i][j][k-1]-
b[i-1][j-1][k]-b[i-1][j][k-1]-b[i][j-1][j-1]+b[i-1][j-1][k-1]+a[i][j][k];

```

我们其实有个更好的办法

```

for(int i=1;i<=n;i++)
  for(int j=1;j<=m;j++)
    for(int k=1;k<=p;k++)
      a[i][j][k]+=a[i-1][j][k];
for(int i=1;i<=n;i++)
  for(int j=1;j<=m;j++)
    for(int k=1;k<=p;k++)
      a[i][j][k]+=a[i][j-1][k];
for(int i=1;i<=n;i++)
  for(int j=1;j<=m;j++)
    for(int k=1;k<=p;k++)
      a[i][j][k]+=a[i][j][k-1];

```

$a_1[i][j][k]=\sum_{l=1}^k a[l][j][k]$ $a_2[i][j][k]=\sum_{l=1}^j a_1[i][l][k]$ $a_3[i][j][k]=\sum_{l=1}^i a[l][j][l]$ a3数组即为所求，更高维度同理
 就是说按维每一维加一遍。这样做可以将复杂度降至 $O(n^t \times t)$ 维数较高的情况下会节省一些时间。

例子

1. 数字列表项目给定一个 $n \times n$ 的矩阵，找一个最大的子矩阵，使得这个子矩阵里面的元素和最大。

最朴素的想法是枚举左下角、右上角并在内部枚举每个元素，复杂度 $O(n^6)$ 利用前缀和降至 $O(n^4)$

好一点的想法是枚举上下边，中间的矩阵就成了一维数列，变成求最大区间和的问题，达到 $O(n^3)$

2.


差分

有时在询问区间和之前有若干对区间的改动。如将 $a[l] \sim a[r]$ 内的数都加上 p 那么我们可以用差分的方法。建立一个数组 b 如果将 $a[l] \sim a[r]$ 内的数加 p 那么在 $b[l]$ 处加 p 在 $b[r+1]$ 处减 p 那么最终 b 的前缀和就是每个位置的增量，再在相应位置修改即可。代码略。

总结

一个基础技巧。

From: <https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link: https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:no_morning_training:%E5%89%8D%E7%BC%80%E5%92%8C&rev=1589379401 

Last update: 2020/05/13 22:16