

前缀和 差分 简单贪心

I. 前缀和(prefix-sum)

前缀和：可以理解为「数列的前 N 项和」，是一种重要的预处理方式。

一维前缀和：对于一个长度为 n 的序列 $\{a_i\}$ ，如果需要多次查询数组里 $[l, r]$ 位置中序列数字的和(即 $\sum_{i=l}^r a_i$)，那么就可以考虑使用前缀和。

我们在高中学习数列的时候，都学过数列的前 N 项和，即

$$S_k = \sum_{i=1}^k a_i$$

这个时候我们知道：

$$S_0 = 0, S_i = S_{i-1} + a_i$$

于是我们可以得到另一个式子：

$$\sum a_{[l,r]} = S_r - S_{l-1}$$

然后我们来考虑一下时间复杂度，看看前缀和究竟带给我们什么了～
让我们先来看看下面这个问题：

对于一个长度为 N 的数列 A ，我们有 Q 次询问
对于每一次询问，我们给出一个区间 $[L, R]$ ，你需要给出

$$\sum_{i=L}^R a_i$$

solution 1: Step1: 读入数列 $A \Rightarrow$ Step2: Q 次循环，每次读入 L, R
 \Rightarrow Step3: $R - L$ 次循环，求和并输出

很显然的是，读入是 $O(N)$ 的，查询的外层循环 Q 次，内层循环 $R - L$ 次，
那么最坏情况就是每一次都枚举 $1 \sim N$ 所有的数字，时间复杂度是
 $O(N + QN)$ 的。

solution 2: Step1: 读入数列 $A \Rightarrow$ Step2: N 次循环预处理出 $S_{i \in [1, N]}$ 前缀和
 \Rightarrow Step3: Q 次循环，每次读入 $L, R \Rightarrow$ Step4: 直接输出 $S_R - S_{L-1}$

读入是 $O(N)$ 的，预处理循环 Q 次，查询的循环 Q 次，对于每一次查询，直接 $O(1)$ 输出，所以复杂度是 $O(N + N + Q)$ 的。

让我们来看看代码吧

```
1  int main() {
2      int n,q,i; scanf("%d%d",&n,&q);
3      int a[n+1]; for(i=1;i<=n;i++) scanf("%d",a+i);
4      for(;q--;){
5          int l,r,ans=0; scanf("%d%d",&l,&r);
6          for(i=l;i<=r;i++) ans+=a[i];
7          printf("%d",ans);
8      }
9  }
```



```
1  int main() {
2      int n,q,i; scanf("%d%d",&n,&q);
3      int a[n+1]; for(i=1;i<=n;i++) scanf("%d",a+i);
4      int pre[n+1]; pre[0]=0;
5      for(i=1;i<=n;i++) pre[i]=pre[i-1]+a[i];
6      for(;q--;){
7          int l,r; scanf("%d%d",&l,&r);
8          printf("%d",pre[r]-pre[l-1]);
9      }
10 }
```



二维前缀和：其实和一维前缀和类似，但有一定区别，实际上，对于一个大小为 $M * N$ 的二维数组 A ，其前缀和的式子应该是

$$S_{i,j} = \sum_{k < i} \sum_{h < j} A_{k,h}$$

那么我们要考虑其递推公式对吧，就应该是这样的：

$$S_{i,j} = A_{i,j} + S(i-1, j) + S(i, j-1) - S(i-1, j-1)$$

就如下图所示，



全部 == 蓝色 + 红绿 + 红黄 - 红

同理可以推导出 $(i, j) \Rightarrow (k, h)$ 的矩阵和为：

$$S_{k,h} - S_{i-1,h} - S_{k,j-1} + S_{i-1,j-1}$$

那么让我们来做一道题吧：

<https://www.luogu.com.cn/problem/P1387>

洛谷 P1387 最大正方形

solution : 最大不包含 0 的正方形，换句话说就是找到最大的正方形区域，使得这个区域内所有数字的和刚好为区域大小，形式化的讲：

$$\sum_{i=x}^{x+l} \sum_{j=y}^{y+l} A_{i,j} = l * l$$

前缀和计算需要 $O(n * m)$ ，枚举边长 l 需要 $O(\min(n, m))$ ，枚举横纵需要 $O(n), O(m)$ ，即 $O(n^3)$ 。

如果不使用前缀和，则计算过程需要多一个 $O(l * l)$ ，最终就是 $O(N^5)$ 的。

所以，显然可以看出前缀和优化了时间复杂度，

关于前缀和我们就先讲到这里！

II. 差分

