斜率优化

主讲人：邓哲也

HDU 3507 Print Article

有一个长度为 N 的正整数序列 A

每次可以连续输出几个数，费用是连续输出的数字和的平方

加上常数 M。

$N \leqslant 500000$

样例输入：

5 5

5 9 5 7 5

样例输出：

230（每个数都单独输出）

HDU 3507 Print Article

设计状态：

用 sum[i] 表示前 i 个数的和。

设 f[i] 为输出前 i 个数的最小费用。

转移方程：

$$f[i] = \min\{f[j] + (sum[i] - sum[j])^2 + M \mid 0 < j < i\}$$

朴素的做法是 $O(n^2)$

HDU 3507 Print Article

$f[i] = \min\{f[j] + (sum[i] - sum[j])^2 + M \mid 0 < j < i\}$

这个算法的时间复杂度瓶颈在哪呢？

计算 f[i] 的时候，要枚举所有可能的转移 j。

而 j 一共有 i 个需要枚举。

如果我们能对所有的可能的转移作一个评估，从最优的那个

位置转移过来，剩下的转移就可以不用枚举了。

HDU 3507 Print Article

$f[i] = \min\{f[j] + (sum[i] - sum[j])^2 + M \mid 0 < j < i\}$

我们可以令 j 取 x 和 y，且 x > y。

如果 x 比 y 要好，那说明：

$f[x]+(sum[i]-sum[x])^2+M < f[y]+(sum[i]-sum[y])^2+M$

$f[x]-2sum[i]sum[x]+sum[x]^2 < f[y]-2sum[i]sum[y]+sum[y]^2$

$(f[x]+sum[x]^2)-(f[y]+sum[y]^2) < 2sum[i](sum[x]-sum[y])$

因为 sum[x] > sum[y]，可以把（sum[x]-sum[y]）除到左边。

HDU 3507 Print Article

$f[i] = \min\{f[j] + (sum[i] - sum[j])^2 + M \mid 0 < j < i\}$

我们可以令 j 取 x 和 y，且 x > y。

如果 x 比 y 要好，那说明：

$$[(f[x]+sum[x]^2)-(f[y]+sum[y]^2)]/(sum[x]-sum[y]) < 2sum[i]$$

我们可以把下标 i 转化为二维平面上的点：

$X = sum[i]$

$Y = f[i] + sum[i]^2$

HDU 3507 Print Article

我们可以把下标 i 转化为二维平面上的点：

$$X = sum[i]$$

$$Y = f[i] + sum[i]^2$$

可以发现这样每个点的 X 坐标是单调递增的。

令 $slope(x,y) = (Y[x] - Y[y]) / (X[x] - X[y])\ (x > y)$

如果 $slope(x,y) < 2sum[i]$，那么 x 比 y 优。

HDU 3507 Print Article

如果 $k < j < i$， 且 $slope(j,k) > slope(i,j)$，那么 j 可以淘汰。

如果 $slope(i,j) < 2sum[i]$，那么 i 比 j 优。

如果 $slope(i,j) > 2sum[i]$，那么有 $slope(j,k) > 2sum[i]$，那么 k 比 j 优。那么 k 比 j 优。

HDU 3507 Print Article

因此每一条线段的斜率应该是递增的。

也就是我们用单调队列维护一个下凸壳即可。

我们要最小化的是：

$F = \min \{f[x]-2sum[i]sum[x]+sum[x]^2\}$

$\Rightarrow \min F = Y - 2sum[i]X$

$\Rightarrow y = 2sum[i]X + F$ 在 y 轴上的截距要越小越好。

也就是我们用一条斜率为 2sum[i] 的直线从下往上，碰到的下凸壳的第一个点就是对应的最优转移。

HDU 3507 Print Article

我们用一条斜率为 2sum[i] 的直线从下往上，碰到的下凸壳的第一个点就是对应的最优转移。

记这个点为 i，上一个点为 p，下一个点为 s。

显然有 slope(i,p) < 2sum[i] < slope(s,i)

由于 2sum[i] 是单调递增的，所以决策具有单调性。

维护一个指针指向每次对应的最优转移即可。

总的时间复杂度是 O(n)

HDU 3507 Print Article

```
double slope(int j, int k){
    int dy = f[j] + sum[j] * sum[j] - f[k] - sum[k] *
sum[k];
    int dx = sum[j] - sum[k];
    if (!dx) return (dy >= 0) ? 1e30: -1e30;
    return (double)dy / (double)dx;
}
```

HDU 3507 Print Article

```
scanf( “%d%d” , &n, &m);
for(int i = 1;i <= n;i ++){
      scanf( “%d” , &sum[i]);
      sum[i] += sum[i - 1];
}
int s = 0, e = 1;
q[0] = 0;
```

HDU 3507 Print Article

```
for(int i = 1;i <= n;i ++){
        while (s + 1 < e && slope(q[s + 1], q[s]) <= 2 * sum[i])
                s ++;
        f[i] = f[q[s]] + m + (sum[i] - sum[q[s]]) * (sum[i] -
sum[q[s]]);
        while (s + 1 < e && slope(i, q[e - 1]) <= slope(q[e - 1], q[e
- 2]))
                e --;
        q[e ++] = i;
}
printf("%d\n", f[n]);
```

下节课再见