对凸包和半平面交问题的研究

尹玉文东 张钰晨 蔡越同 李灏冬

指导教师：张敏

2022.05.10

信息技术的普及与推广

计算机网络的飞速发展日益改变这人们的生活和学习方式。从全球范围来看，信息化教育的改革浪潮一浪高过一浪，发展信息化教育是大势所趋。

随着信息化的建设，CSP、NOIP等算法竞赛在我国中学生群体的影响力与日俱增。

动态规划的应用广泛

近年来，动态规划在信息学中的应用愈发广泛。以算法竞赛为例，越来越多的竞赛中包含了动态规划的题目。

算法层面优化效果极佳

对于一些时间复杂度本身较劣的 DP 思路，硬件运行速度的提升往往是常量级的，在数据规模快速增长时效果甚微。

- DP：Dynamic Programming,动态规划。
- LIS：Longest Increasing Subsequence，最长上升子序列。
- TSP：Traveling Salesman Problem，旅行商问题。

  假设有一个旅行商人要拜访n个城市，他必须选择所要走的路径，路径的限制是每个城市只能拜访一次，而且最后要回到原来出发的城市。路径的选择目标是要求得的路径路程为所有路径之中的最小值。

- 结题报告中 DP 状态，有如下表达形式：[]、（）、和下角标。

基于状态压缩的动态规划

- 状态压缩：将一个较小的集合内每个元素的状态通过特定的计算方法转换为单个整数。
- 位运算

  与、或、取反、异或、左移、右移
- 位运算技巧

  交集、并集、补集、取位、判断子集
- 适用范围：对于一类需要记录整个集合内每个元素状态的问题，状态压缩优化状态是更优的选择。

按数位顺序进行的动态规划

- 技巧性较强。
- 适用问题类型：
  - 给定的条件一般与数的组成有关。
  - 数据规模一般较大
  - 最终目的为计数
- 状态转移技巧：

  前缀和思想处理。

  使用记忆化搜素，代替循环。

单调队列优化动态规划

往往是题目中有一些条件使得在没有该条件时的最优决策不合法了，因此使用单调队列排除不可能在接下来成为最优决策点的元素，保留其余元素。

四边形不等式优化动态规划

利用四边形不等式的性质，对于一些具有决策单调性的动态规划问题，可以利用单调性进行优化。

斜率优化动态规划

斜率优化可以在将转移式变形后，通过考察几何意义，将可能成为最优转移点的集合缩小到一个凸包上，再根据题目条件进行优化。

CDQ 分治优化动态规划

CDQ分治能对可离线的高维偏序问题以及转移包括高维偏序条件的动态规划问题优化其时间复杂度。

以**第五章：斜率优化动态规划**为例。

算法概述

斜率优化可以在将转移式变形后，通过考察几何意义，将可能成为最优转移点的集合缩小到一个凸包上，再根据题目条件将转移的时间复杂度缩小到 $\mathcal{O}(n)$ 或 $\mathcal{O}(n\log_2 n)$ 。

应用场景

可以应用斜率优化的动态规划模型往往某一维度的状态数为 $\mathcal{O}(n)$ 级别，而为找到最优转移点，单次的状态转移需考察 $\mathcal{O}(n)$ 个子阶段，这使该维度的转移的时间复杂度开销达到 $\mathcal{O}(n^2)$ 。

应用实例

题目名称：玩具装箱

题目来源：NOI2008 湖南省省队选拔活动

有 n 个玩具，第 i 个玩具价值为 c_i。要求将这 n 个玩具排成一排，分成若干段。

对于一段 $[l,r]$，它的代价为：

$$\left(r-l+\sum_{i=l}^{r}c_i-L\right)^2$$

求分段的最小代价。

数据范围：

$$1\le n\le 5\times 10^4, 1\le L, 0\le c_i\le 10^7$$

令 f_i 表示前 i 个物品，分若干段的最小代价。有状态转移方程：

$$f_i = \min_{j<i}\{f_j + (pre_i - pre_j + i - j - 1 - L)^2\}$$

其中 $pre_i = \sum_{j=1}^{i} c_j$。直接转移时间复杂度为 $O(n^2)$，无法解决本题。

为简化状态转移方程式，令 $s_i = pre_i + i, L' = L + 1$，则：

$$f_i = \min_{j<i}\{f_j + (s_i - s_j - L')^2\}$$

设 j 为使 f_i 最小的转移点，有：

$$f_i = f_j + (s_i - s_j - L')^2$$

$$f_i = f_j + (s_i - s_j - L')^2$$

考虑一次函数的斜截式 $y = kx + b$，将方程转化为这个形式。

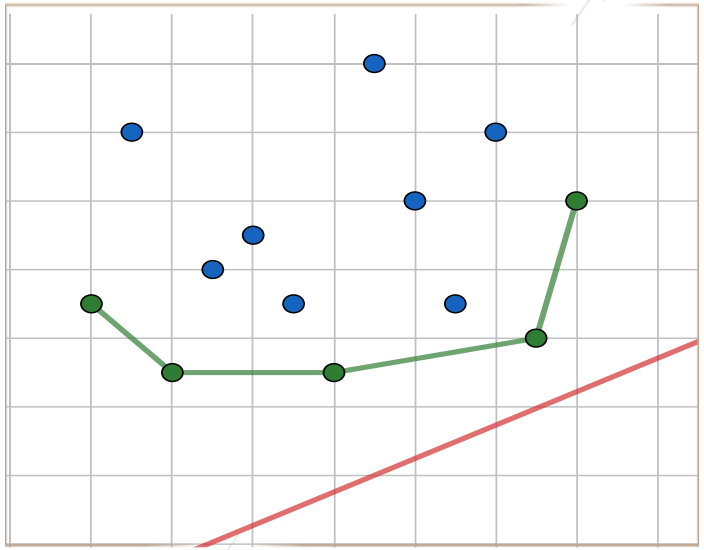
其中变量 x, y 与 j 有关，b, k 与 i 有关，且要求 x 随 j 单调递增，仅 b 中包含 f_i。

按照上面的规则，对方程进行整理得到：

$$f_j + (s_j + L')^2 = 2s_i(s_j + L) + f_i - {s_i}^2$$

$$\begin{cases} y = f_j + (s_j + L')^2 \\ k = 2s_i \\ x = s_j + L \\ b = f_i - {s_i}^2 \end{cases}$$

$$\begin{cases} y = f_j + (s_j + L')^2 \\ k = 2s_i \\ x = s_j + L \\ b = f_i - s_i^2 \end{cases}$$



(x_j, y_j) 的几何意义为：直线 $y = kx + b$ 上的一个点，又因为转移时目的是最小化 f_i，在上面的表示当中，f_i 只与直线的截距 b 有关。

所以问题可转化为如何选取 j，使得过点 (x_j, y_j) 的直线的截距 b 最小。

注意到直线的斜率不变，相当于平移直线 $y = kx$，直到其经过图中的一个点。

于是可以在转移的同时维护 (x_i, y_i) 构成的凸包，利用单调性二分得到时间复杂度为 $O(n\log n)$ 的算法。

发现： $k = 2s_i$ 关于 i 单调递增，因此可以配合单调队列得到时间复杂度为 $O(n)$ 的算法。

按照如下转移方程进行即可：

$$f_i = f_j + (s_i - s_j - L')^2$$

```
const int MAXN = 5e4 + 10;
int n, L;
int c[MAXN], s[MAXN], f[MAXN];
double slope(int i, int j) {
            return ((f[j]+(s[j]+L)*(s[j]+L)) - (f[i]+(s[i]+L)*(s[i]+L))) / (double)(s[j] - s[i]);
}
int head, tail, q[MAXN];
signed main() {
            scanf("%lld%lld", &n, &L);
            L += 1;
            for(int i = 1; i <= n; i++) {
                        scanf("%lld", &c[i]);
                        s[i] = s[i - 1] + c[i] + 1;
            }
            head = tail = 1;
            for(int i = 1; i <= n; i++) {
                        while(head < tail && slope(q[head], q[head + 1]) <= 2 * s[i]) {
            head++;
        }
                        f[i] = f[q[head]] + (s[i] - s[q[head]] - L) * (s[i] - s[q[head]] - L);
                        while(head < tail && slope(q[tail - 1], q[tail]) >= slope(q[tail], i)) {
            tail--;
        }
                        q[++tail] = i;
            }
            printf("%lld", f[n]);
            return 0;
}
```

总结与展望

- 知识覆盖面广
- Latex语言排版
- C++程序完整实现
- 算法竞赛中的典例

创新

不足

- 创新性有待提高
- 研究具有局限性

致谢 Thanks

感谢聆听。

感谢张敏老师。

感谢年级与学校对研究性学习的支持。