2020 年 第 5 問

ジャンル: 算法
難易度: Lunatic

> **📖問題の概要**
> 非常に難しい。作問者は (1-2) を誘導にしたかったのだと思うが、誘導が雑なため、普通の人は正しく誘導されず、まちがった方針に誘導されてしまうだろう。目的関数が二次関数なので解析的に頑張れてしまうことが、(1-2) の誘導が誘導になっていない原因である。算法の問題であることを意識して解こう。
>
> これを解いても勉強にはならないが、興味がある人は見てみればいいだろう。ちなみに、この問題は Isotonic Regression と呼ばれる問題で多くの派生問題があり、50 年代から研究されている。SOSA2019 というシンポジウムでこの問題を DP で解く論文が発表され、それがそのまま 2019 年 8 月の試験問題になっている。
>
> 競技プログラミングをかなりやっている人ならば、この問題を知っている人がいるかもしれない。競技プログラミングの文脈では slope trick と呼ばれている。

以降、以下の記号を用いる。

$$d_n(A,B)=\sum_{i=1}^{n}(A[i]-B[i])^2$$

(1-1)

まず、この問題の解は一意であることを示す。

$$\begin{aligned}\underset{B_1,\dots,B_n}{\text{minimize}} \quad & \sum_{i=1}^{n}(A_i-B_i)^2\\ \text{subject to} \quad & B_1\le B_2\le\cdots\le B_n\end{aligned}$$

は以下のように狭義凸な二次計画問題 (Convex Quadratic Programming) として定式化できる。

$$\begin{aligned}\underset{B_1,\dots,B_n}{\text{minimize}} \quad & \frac{1}{2}\boldsymbol{b}^T I\boldsymbol{b}-2\boldsymbol{a}^T\boldsymbol{b}\\ \text{subject to} \quad & \begin{pmatrix}-1&1&0&0&\cdots&0\\0&-1&1&0&\cdots&0\\0&0&-1&1&\cdots&0\\\vdots&&&&\ddots&\\0&0&\cdots&0&-1&1\end{pmatrix}\boldsymbol{b}\le\begin{pmatrix}0\\0\\\vdots\\0\\0\end{pmatrix}\end{aligned}$$

実行可能領域は多面体より凸集合を成す。目的関数は正定値対称 $I\succ O$ より、目的関数は凸関数。よって最適解は一意である。最適解が一意であることがわかったので、最適性の十分性だけ示す。

問題文で与えられている単調非減少配列を C とする。すなわち、$C[i]=B[i]\quad(1\le i\le n), C[n+1]=A'[n+1]$ とする。このとき、$d_{n+1}(A',C)=d_n(A,B)$ である。一方で、A' に対する最適な単調非減少配列が D で、C は最適ではない $d_{n+1}(A',D)<d_{n+1}(A',C)$ とすると、

$$d_n(A,D\text{ から末尾を削ったもの})\le d_{n+1}(A',D)<d_{n+1}(A',C)=d_n(A,B)$$

これは B の A に対する最適性に矛盾する。よって C は最適である。

(1-2)

解析的な証明を与える。この証明は重要でなく、もっと簡潔で (2) と関連する証明があるので読み飛ばして構わない。この証明を読んだところで (2) は解けない。

記号を定義する。

$$a_r \coloneqq \frac{1}{r}\sum_{i=1}^{r} A'[i] \quad (\text{先頭 } r \text{ 個の平均})$$

いくつか補題を示す。

Lemma 1: A に対する近似配列 B が、ある r $(1 \leq r \leq n)$ で、$B[1] = \cdots = B[r]$ の制約があり、$B[r+1]$ 以降の要素は既に決まっているとする。
このとき、$B[r+1] \geq a_r$ ならば、$B[1] = \cdots = B[r] = a_r$ とするのが最適である。 $B[r+1] < a_r$ ならば、$B[1] = \cdots = B[r] = B[r+1]$ とするのが最適である。

Proof: $d_r(A, c) = \sum_{i=1}^{r} (A[i] - c)^2$ を c で微分することで、最小化する定数 c は a_r であることがわかる。$d_r(A, c)$ が c に対する二次関数なので、$B[r+1] < a_r$ の場合もできるだけ a_r に近い場所が最適である。これは $c = B[r+1]$ を意味する。 □

Lemma 2: 長さ n の配列 A に対して、最適な近似配列 B が $B[1] = \cdots = B[n]$ であるとする。
このとき、$a_r \geq a_n$ $(1 \leq r \leq n)$

Proof: Lemma 1 より、$B[1] = \cdots = B[n] = a_n$ である。もし、$B[1] = \cdots = B[r] = c \neq a_r$ と変更しても、B の最適性より悪化しかしない。Lemma 1 より、$a_r < a_n$ ならば $c = a_r$ とすると改善してしまうので、$a_r \geq a_n$ である。 □

準備は終わったので、示していく。

Theorem 1: $B[1] = \cdots = B[n]$ かつ $A'[n+1] < B[n]$ とする。このとき、$B'[1] = \cdots = B'[n+1]$ かつ $B'[n+1] < B[n]$

Proof: Lemma 1 より、$B[1] = \cdots = B[n] = a_n$ である。さらに問題文より、

$$b_{n+1} = \frac{1}{n+1}A'[n+1] + \frac{n}{n+1}a_n < \frac{1}{n+1}B[n] + \frac{n}{n+1}a_n = a_n$$

であり、Lemma 2 と合わせて、$a_r \geq a_n > a_{n+1}$ $(1 \leq r \leq n)$。

さて、B' の最後の変化点について考える。$B'[1] = \cdots = B'[r] < B'[r+1]$ なる最後(最初?)の変化点 r $(1 \leq r \leq n-1)$ が存在するとする。Lemma 1 より、変化点が存在するならば、そこでは $B'[1] = \cdots = B'[r] = a_r < B'[r+1]$ でなくてはいけない。B' は単調非減少なので、$B'[r+1] \leq B'[n] \leq B'[n+1]$。 よって、

$$B'[n+1] \geq B'[n] \geq B'[r+1] > a_r \geq a_n$$

だが、これは B の最適性に反する。長さ n の近似配列は $B[i] = a_n$ が最適で、$B'[n+1] > a_n$ なので実行可能解に入っている。よって矛盾。$r = n$ か $r = n+1$ であるが、$r = n$ のときは明らかに $B'[n+1]$ をより小さくして $B'[n+1] = B'[n]$ にしたほうが改善するので、$r = n$ ではない。よって、$r = n+1$ であり、$B'[1] = \cdots = B'[n+1]$。Lemma 1 より、$B'[1] = \cdots = B'[n+1] = a_{n+1} < a_n = B[n]$ より、示された。 □

(2)

以下の議論は全て参考文献の論文の受け売りなので、わからないところがあれば参考文献を読んでみるといいだろう。

さて、まずは区分二次凸関数(piecewise quadratic convex function)という概念を導入する。区分二次凸関数とは、関数が有限個の全ての区間(非有界な区間でも可)において二次関数になっており、全体で見ると凸関数である関数のことである。 定義から、区分二次関数に二次関数を足しても区分二次関数であり、さらに、区分二次凸関数に凸な二次関数を足しても区分二次凸関数である。

では、以下の部分問題を考える。

$$f_k(x) = \min\left\{ \sum_{i=1}^{k} (B_i - A_i)^2 \;\middle|\; B_1 \le B_2 \le \cdots \le B_k = x \right\}$$

定義から簡単に以下がわかる。

$$f_k(x) := \min\{f_{k-1}(z) \mid z \le x\} + (x - A_k)^2$$
$$f_1(x) := (x - A_1)^2$$

Lemma 3:
1. f_k は区分二次凸関数である
2. f_k の区間は最大 k 個
3. 最も右の区間の二次関数は $x^2 - 2A_k x + c$ (c は適当な定数) で、最も左の区間の二次関数は $kx^2 - 2\left(\sum_{i=1}^{k} A_i\right)x + c'$ (c' は適当な定数)

Proof:

$$g_{k-1}(x) := \min\{f_{k-1}(z) \mid z \le x\}$$

と定義する。 補題の主張は $k = 1$ で自明に成り立つ。f_{k-1} で成り立つとする。すると凸性より、$f_k(x)$ は最初は単調減少し、ある点からは単調増加する。p_{k-1} をそのような $f_{k-1}(x)$ の最小値を与える点とする。すると、$g_{k-1}(x)$ は、$x \le p_{k-1}$ で $z = x$ が最適で、$g_{k-1}(x) = f_{k-1}(x)$。$x \ge p_{k-1}$ では $z = p_{k-1}$ が最適であり、$g_{k-1}(x) = f_{k-1}(p_{k-1})$。 よって、 $g_{k-1}(x)$ は $f_{k-1}(x)$ と同じ単調減少する区間を持ち、単調増加する区間は全て定数関数 $f_{k-1}(p_{k-1})$ に置き換えられている。つまり、g_{k-1} は最大 k 個の区間を持つ、区分二次な広義凸関数である。 $f_k(x) = g_k(x) + (x - A_k)^2$ より、$f_k(x)$ は最大 k 個の区間を持つ区分二次凸関数である (区間ごとの二次関数の形に $x^2 - 2A_k x + A_k^2$ が足されるため)。最も右の区間は、$g_{k-1}(x)$ の最右区間が定数関数だったことから従う。最も左の区間も同様に、帰納法の仮定と $g_{k-1}(x)$ の形から従う。 □

証明より、問題の解は $B_n = p_n, B_{k-1} = \min(p_{k-1}, B_k)$ と構成することができることに注意する。

ここまで示したことを図でもう一度説明する。

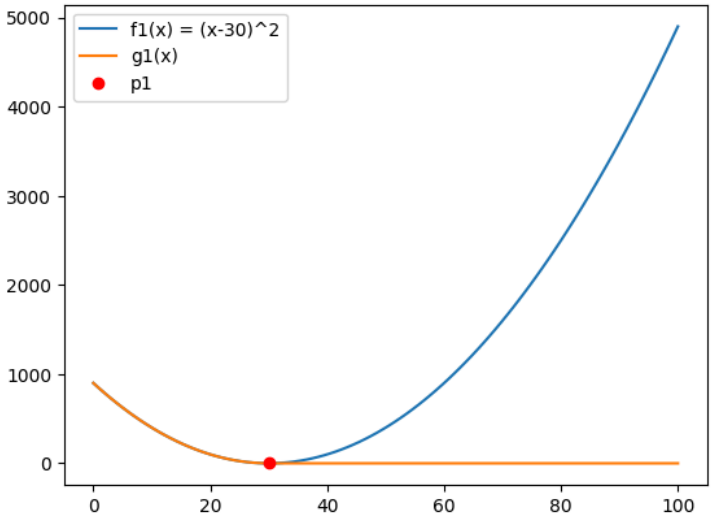

Figure 1: $f_1(x) = (x-30)^2$ のときの $f_1(x), g_1(x), p_1$ の例

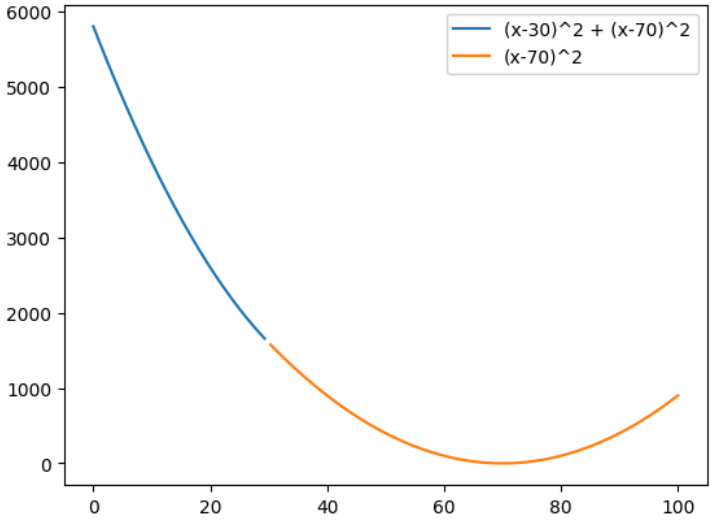

Figure 2: $f_2(x) = g_1(x) + (x-70)^2$ の例。各区間ごとに二次関数を足せばよい。元々全ての区間は二次関数なので、足した結果、すべての区間は二次関数。また、g_{k-1} は凸なので凸関数である $(x-70)^2$ を足した f_k も凸であることがわかる。

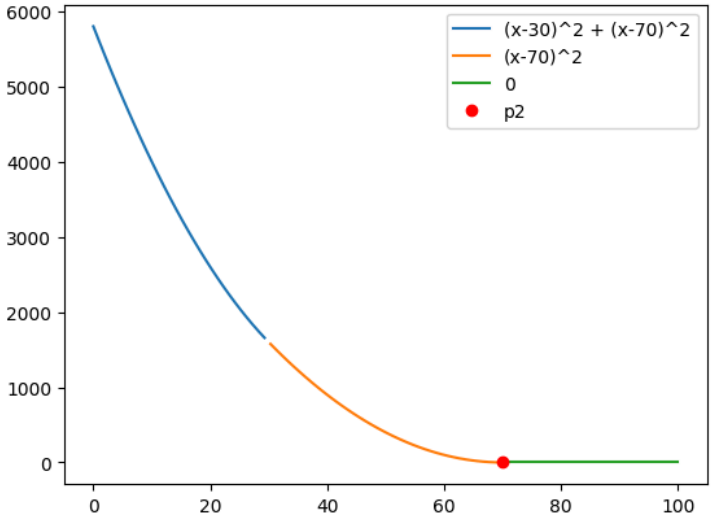

Figure 3: $g_2(x)$ の例。

$A_1 = 30, A_2 = 70$ の例だが、$A_3 = 50$ であれば f_3, g_3 はどのようになるか？p_1, p_2 とは違い、p_3 は最も右の区間に現れるとは限らないであろう。 さて、これで基本的な説明は終わった。 次の問題は、この区間とそこでの関数をどのように表現するかである。区間は被りがないので、各区間の端点とその区間での二次関数の係数 $ax^2 + bx + c$ の (a, b, c) を保存すればよい。二次関数が足されるときはそのまま係数同士を足せばよいし、最小な点 p_k を求めるときは、各区間の二次関数の最小点 $x = -\frac{b}{2a}$ を求めて、それが区間内に入っていればそれが最小点である。入っていなければ、隣の区間を順番に探索していけばよい。

それではアルゴリズムを与える。

Algorithm 1

Q := ∅; // 区間の左端をキーとする優先度付きキュー
Q.insert($(-\infty, \infty), (0, 0, 0)$); // 区間 $(-\infty, \infty)$ は $0x^2 + 0x + 0$
for $k = 1, ..., n$ **do**
  // g_{k-1} から開始
  for all q ∈ Q **do**
    q の係数に $x^2 - 2A_k x + A_k^2$ を足す;
  Obj := Q.findmax; // 最も右の区間から見ていく
  while $-\frac{\text{Obj.b}}{2\ \text{Obj.a}} <$ Obj.left **do**
    Q.deletemax;
    Obj := Q.findmax;
  $p_k := -\frac{\text{Obj.b}}{2\ \text{Obj.a}}$
  if Obj.left $\neq p_k$ **do** // 幅 0 の区間が生まれるのを防ぐ
    Obj.right := p_k;
    Q.insert($(p_k, \infty), (0, 0, \text{Obj.a}\ p_k^2 + \text{Obj.b}\ p_k + \text{Obj.c})$); // p_k より右は定数関数
  else
    Obj := $((p_k, \infty), (0, 0, \text{Obj.a}\ p_k^2 + \text{Obj.b}\ p_k + \text{Obj.c}))$;
// B_k を求めていく
$B_n := p_n$

Algorithm 1

for $k := n-1, n-2, ..., 1$ **do**
  $B_k := \min(p_k, B_{k+1})$

以下は補足であるが、実はこのアルゴリズムはさらに高速化できる。まず、優先度付きキューはスタックに変更できる。何故なら区間を右側から順番に見ていくか、最も右の区間に定数関数を追加することしかしていない。

次に、区分二次関数を表現する方法について考える。区間内における定数項 c については p_k を求めるときに関与しない。よって持つ係数は実は a, b のみでよい。さらに、区間の係数を持つのではなく、区間の区分点において、左右の係数の差を持つようにすればよい。こうすると区間に二次関数を足すとき、最も右の区間の係数を増やすだけでよくなる。これは slope trick と呼ばれるテクニックで、競技プログラミングでよく使われる。しかも、 このような高速化をすると全体で $O(n)$ になる。とはいえこれらのテクニックを使わず愚直に実装しても $O(n^2)$ で多項式時間なので、問題に解答する分には十分であるが、一応紹介しておいた。

今回は DP による解法について書いたが、それ以外の解法も実に多様な手法が提案されている。(2) は二次凸計画問題として定式化できるので楕円体法で多項式時間で解けます！ w でも解答としてはいいかもしれない...。最適化の理論は汎用的で強いですね。

(1-2) 別解

Lemma 3 の証明を読めば自明であると思うが、$B[n] = p_n, B[k-1] = \min(p_{k-1}, B[k])$ を使うと、$B[1] = B[1] = \cdots = B[n]$ ならば、$p_k \geq B[n]$ であることがわかる。よって、g_n の左の区間が $nx^2 - 2\left(\sum_i A_i\right)x + \sum_i A_i^2$、右の区間が定数。となっている。ここに $\left(x - A_{n+1}\right)^2$ を足すと、定数の区間では最小値を取らないことが自明にわかる。よって $p_{n+1} < p_n$ であるので、$B'[1] = \cdots = B'[n+1] < B[n]$ と簡潔に示せる。

間違った解法について

drive や github に上がっている、B をシフトして平らにしたあと (1-2) を適用して、その後戻す。というアルゴリズムは間違っている。B を単調性を持ったシフトをして平らにして、(1-2) を適用するところまでは論理として正しいが、その後戻す部分で最適性が失われてしまう。単調性を持ったシフトに対して最適性は保存されるが、逆のシフトでは失われることに注意する。

参考文献

[1] G. Rote, “Isotonic regression by dynamic programming,” in *2nd Symposium on Simplicity in Algorithms (SOSA 2019)*, 2019.