1 CHT で高速化する DP

以下のような遷移をする DP を考える．

Algorithm 1: 愚直な DP

$dp[0] \leftarrow c$ ▷ 初期条件は計算できるとする．

for $i \in \{1, \ldots, N-1\}$ **do**

  $dp[i] \leftarrow \min_{0 \leqslant j < i} \{p(j) \cdot q(i) + r(j)\} + s(i)$

$p(j)$ や $r(j)$ は $dp[j]$ を含む式でもいいし，関係ない式でも問題ない．

ここで，直線の集合に関する以下の処理をできるデータ構造を用意する．これは convex hull trick などと呼ばれるものである．

- 直線 $y = ax + b$ を追加する
- 管理している直線のうち，$x = x_0$ での y の最小値を答える

これを用いて上の DP を以下のように高速化できる．直線の集合 S をこのデータ構造を用いて管理する．

Algorithm 2: CHT で高速化した DP

$dp[0] \leftarrow c$

$S \leftarrow \{\}$

for $i \in \{1, \ldots, N-1\}$ **do**

  $S \leftarrow S \cup \{p(i-1) \cdot x + r(i-1)\}$

  $dp[i] \leftarrow \min_{ax+b \in S} \{a \cdot q(i) + b\} + s(i)$

このような遷移に変形させることができれば勝ちなので，それを意識して式変形するとよい．典型的には，以下のような遷移をする DP などがこの枠組みに当てはまる．

$$\begin{aligned} dp[i] &= \min_{0 \leqslant j < i} \{dp[j] + (i-j)^2 + C\} \\ &= \min_{0 \leqslant j < i} \{dp[j] + i^2 - 2ij + j^2 + C\} \\ &= \min_{0 \leqslant j < i} \{\underbrace{(-2j)}_{p(j)} \cdot \underbrace{i}_{q(i)} + \underbrace{(j^2 + dp[j])}_{r(j)}\} + \underbrace{(i^2 + C)}_{s(i)}. \end{aligned}$$

convex hull trick の実装として Li–Chao tree を用いる場合などは，$q(i)$ の取りうる最大値が小さくなるように変形すると，クエリ先読みなどの必要がなくなることがあったりしてうれしそう．もっとも，`std::map` などを利用してオンラインで処理することもできる．