ARC056D サケノミ
解説

https://atcoder.jp/contests/arc056/tasks/arc056_d

目次

問題概要

問題

N 個のグラスがあり，最初は全て空である．t_{ij} $(1 \leq j \leq M_i)$ 時間後に i 番目のグラスが空であれば，i 番目のグラスにドリンク i が注がれる．また，各ドリンクには美味しさ w_i が決まっている．好きな時刻に以下の行動を好きな回数だけ行える時，飲むドリンクの美味しさの総和を最大化せよ．

- 全てのグラスのドリンクを飲み干す

制約

- $1 \leq N \leq 5 \times 10^5$
- $\sum M_i \leq 5 \times 10^5$
- $-10^9 \leq w_i \leq 10^9$

問題把握

この問題のポイント
美味しさが負のドリンクがあること（全部正なら自明）．負のドリンクがあることにより，今はドリンクを飲まないという選択が最適になる可能性がある．
(例)
ドリンク１：美味しさ 100
ドリンク２：美味しさ -10
ドリンクが注がれる順番が，$2 \to 2 \to 1 \to 2 \to 1$ であれば，最初の 1 が注がれた直後に飲むとダメ

問題把握

今回のように和の最大化（最小化）問題で，複数の条件がごちゃごちゃ絡み合う系の問題では，基本的に貪欲が効かない（例：ナップザック問題，例外：最小全域木（より一般にマトロイド））.
なので，このような問題では，どのように選ぶのが最適かを考察するのは無意味なことがほとんど（ナップザックでこのような考察をして失敗した経験がある人も多いはず）.
$\rightarrow$ このような問題では，まず初めはどのように dp やシミュレーションするかを考えるのが無難.

ヒント１，２

ヒント

時間軸に沿って dp することが目標です．

どのような dp をするかは後述するが，素直に思いつく dp を考える．

ヒント１，２

ヒント

時間軸に沿って dp することが目標です．

どのような dp をするかは後述するが，素直に思いつく dp を考える．

ヒント

i 番目のグラスに時刻 $t_{i,j}$ に注がれるドリンクを飲むための条件を考えてみましょう．

これは得点に関わる情報．
dp において，何の情報を持つ必要があるかに直接影響するのでしっかり考える必要がある．

ヒント 3

ヒント

i 番目のグラスに時刻 $t_{i,j}$ に注がれるドリンクを飲むのは，

- $t_{i,j-1} < t < t_{i,j}$ なる時刻 t にドリンクを飲み干した．
- $t_{i,j} < t$ なる時刻 t にドリンクを飲み干した．

の 2 条件で表すことができます．

ヒント 3

ヒント

i 番目のグラスに時刻 $t_{i,j}$ に注がれるドリンクを飲むのは，

- $t_{i,j-1} < t < t_{i,j}$ なる時刻 t にドリンクを飲み干した．
- $t_{i,j} < t$ なる時刻 t にドリンクを飲み干した．

の 2 条件で表すことができます．

1 つ目の条件は，そもそも時刻 $t_{i,j}$ にドリンクが注がれるための条件．
最後にドリンクを飲み干した時刻が分かれば，今新たに飲もうとするときに上記の条件をみたす $t_{i,j}$ が存在するかを判定する情報は十分
($t_{i,j-1} < t_{last} < t_{i,j} < t_{now}$ かどうかみたいな話になる)．

ヒント 4

ヒント

$dp_i =$ 最後に時刻 i にドリンクを飲み干したときの最高得点 という dp を考えます．このとき，更新式は，

$$dp_j = \max_{i<j}(dp_i + \text{ヒント 3 の条件を満たす } t_{k,l} \text{ が存在するような } k \text{ における } w_k \text{ の総和})$$

となります．ここで寄与分解を考えます．すなわち，各 $t_{k,l}$ について，そいつが理由で w_k が足される dp_i はどのような分布をしているか考えてみましょう．

dp の遷移の詳細

$dp_i =$ 最後に時刻 i にドリンクを飲み干したときの最高得点

dp_i から dp_j に遷移することを考える（つまり，最後にドリンクを飲み干したのが時刻 i の状態で，次に飲み干すのが時刻 j という状況）.

このとき，時刻 i 以降，時刻 j 以前に注がれるドリンクの分だけ得点が変化する.

ドリンク k が飲まれるのは，$i < t_{k,l} < j$ なる $t_{k,l}$ があるとき.

dp の遷移の詳細

ドリンク k が飲まれるのは，$i < t_{k,l} < j$ なる $t_{k,l}$ があるとき．
しかし，これを素直に実装すると，$i < t_{k,l} < t_{k,l+1} < j$ であった時に，
ドリンク k を 2 回数えてしまう！
これを，ヒント 3 のように，**いつ注がれたドリンク k を飲むのか考える**ことによって重複を除く．

dp の遷移の詳細

ドリンク k が飲まれるのは，$i < t_{k,l} < j$ なる $t_{k,l}$ があるとき．
しかし，これを素直に実装すると，$i < t_{k,l} < t_{k,l+1} < j$ であった時に，
ドリンク k を 2 回数えてしまう！
これを，ヒント 3 のように，**いつ注がれたドリンク k を飲むのか考える**
ことによって重複を除く．
時刻 i から時刻 j への遷移で時刻 $t_{k,l}$ に注がれたドリンク k を飲む
$\Leftrightarrow t_{k,l-1} < i < t_{k,l} < j$ を満たす．
このようにすると重複の問題はなくなる．

解法

この段階で $O((\max t_{i,j})^2)$ で解ける．
上記 dp を寄与分解によって高速化する．
すなわち，時刻 $t_{k,l}$ に注がれるドリンク k が得点に加えられるのは，どのような $i \to j$ の遷移か？ ということを考える．ここで今までの考察から，

- i の条件：$t_{k,l-1} < i < t_{k,l}$
- j の条件：$t_{k,l} < j$

ということが分かる．

解法

- i の条件：$t_{k,l-1} < i < t_{k,l}$
- j の条件：$t_{k,l} < j$

これらの条件は i と j で独立である．また，時間軸に沿って処理することを考えれば，j の条件は時刻 $t_{k,l}$ の後に処理する j 全てについて勝手に満たされる．

よって，時刻 $t_{k,l}$ 以降に上記の条件を満たす i から遷移する際に，時刻 $t_{k,l}$ に注がれるドリンク k が得点に影響することが分かる．

解法

遷移の式をもう一度書く．

$$dp_j = \max_{i<j} \left(dp_i + \sum_{\text{条件}} w_k \right)$$

ここで，w_k が足される条件は i に関するもののみであり，$t_{k,l}$ によって条件を満たす i は $t_{k,l-1} < i < t_{k,l}$ を満たす i である．
よってこれは，$t_{k,l-1} < i < t_{k,l}$ を満たす i に対し，時刻 $t_{k,l}$ の処理で，$dp_i \leftarrow dp_i + w_k$ と更新することで実現できる．
dp の遷移式から，区間可算，区間 $\max$ ができるデータ構造で実現できることが分かる（遅延セグメント木，starry sky tree など）．