上下に動く水面

原案：野田
解答：牟田、三廻部、高橋、安達

解答状況

正解チーム数：3
最速正解：141分48秒（USAGI Code）

実は DP でした.

ジャッジの出題意図としては Event Driven な問題のつもりで出題していました.

しかし、実は DP で解ける問題でした.

ごめんなさい.

DP で解くには

- 閘門とゴールは全て整数座標，船の間隔も整数なので
  - 整数座標に各船が何時に入って，何時に出るのか
  - 各閘門は何時に利用可能になるのか
- 非整数の場所でどう動くかを考慮する必要はありません.
- 先頭の船から順に上記の情報を埋めていけば解けます.

実装すべきソースコードの量は少ないので考えてみてください

以下は Event Driven の解説

Event Driven って何？という人,
解法が思いつかなかった人は読んでみてください.

時間を愚直に進める方法

許容誤差が 10^{-6} だから、10^{-6} 刻みで時刻を増やしていけば,
解けますか？

無理です.

川の全長が 1000 船の速度が 1 の場合を考えると答えの時間はざっと 1000. 先程の刻み幅 10^{-6} で割ると 10^{9} 回ループを回す必要があります. どうしてもTimeLimitExceeded になります.

時間をスマートに進める方法

> 船は一瞬で任意の船速に達し,
> さらに一瞬で静止することすらできる.

という前提があるので細かく時間を進めるというケチケチしたことをせずに一気に興味のあるイベントが起こる時間まで進めてしまいましょう.

ここでのイベントとは

- 閘門の注水・排水
- 船の閘門への到着

などがあります.

どれだけ時間をすすめr

闇雲にイベントを選んで時間を進めると重要な出来事を見落とすかもしれません

例えば

- 船が閘門に衝突する
- 閘門の注／排水が終わったのに船が閘門に入らない／出ない

毎回最小のイベントを選択が大変

Java なら PriorityQueue を
C++ なら priority_queue を使いましょう.

priority queue は要素の挿入と最小の要素の取得が O (log N) で
できるでーたこ

シュミレーション問題は実装が大変

- 手作業を大事にしてください.
- サンプル入力が完全とは限りません、簡単な例を作って自分のコードの出力と手作業の結果を比較しましょう.
- また、途中出力をダンプしてどこから計算が狂ったかを見るのも大事です。

やっぱり
シュミレーション問題は実装が大変

頑張ってください