第 35 回競技部門：30058

タイトル：寿司が好きなコンピュータ，何か知ってる？
「データ巻き」ってね

学校名：神戸市立工業高等専門学校

1. はじめに

今回の型抜きパズルは，抜き型の種類や使用する位置など，非常に分岐が多いものとなっている．そこで，私たちは後述する基本の動きを一般化し，そこから考えられる最適解を見つけるような戦略を取ることとした．

2. アプローチ

2.1 大まかな流れ

今回作成した基本の動きでは，主に行と列を別々に揃えていくような戦略となっている．その流れを以下に示す．

(1) 最終盤面を下の行から順に見ていき，各行の 0-3 の各要素の数を揃える
  (a) はじめに，縦方向の寄せのみで過剰な要素を不足した要素で置き換えていく
  (b) その後，残った過剰な要素のピースを，そのピースに近い列にある不足した要素を置き換えていく
(2) 最終盤面を右の列から順に見ていき，各列を横方向の寄せのみで揃える

2.2 ピースの移動法

ピースの移動については，移動させたい位置までに存在するピースを抜いて移動させる場合を考える．定型抜き型のみの使用では，移動させたい距離をNと置くと必要手数は$O(\log N)$手となる．しかし，盤面の端に移動させるだけの場合は，移動させたいピースを抜いて移動させると，1 手で済む．これを利用し，各行・列を揃える際に一番下の行・一番右の列に揃え，反対側に移動させていくことで，少ない手数で盤面を揃えていくことができる．

また，(1.b)では，揃えている行と平行に移動させる必要がある．ここでは盤面の端に揃えるということは難しいので，先ほど述べた$O(\log N)$手で揃えることになる．しかし，揃えている途中の一番下の行を回避して型抜きをしていく必要がある．そこで，定型抜き型のタイプII を使用することで，盤面が崩れてしまうことを防ぐ．

2.3 複数パターンの流れ

前述した流れは，下の行から右の列という順番の流れになっている．しかし，最初にどの方向から始めるか，その後どちらの横方向から揃えていくかという，計 8 通りの分岐がある．そのそれぞれのパターンを検証することにより，最適な流れを見つけることができる．これを実現するために，初めから盤面をそれぞれ反転させた状態で同じような流れを並列処理で行う．

2.4 複数種の抜型の使用

なんといっても，適切な抜き型を使用することが最も手数を削減することができるカギとなる．そこで，使用しても盤面の崩れない定型抜き型を使用したときのそれぞれの評価値を過剰な要素数などから判断することにした．その評価値を比較し，最適な抜き型の種類を決定する．

3. システム

全体を JavaScript で構築し，ソルバーと UI でそれぞれの実装を分割している．

3.1 ソルバー

Promise を利用した非同期処理をベースに，並列化のため，アルゴリズム部を WebWorker に分割してメインスレッドから分断している．

具体的な処理形態は，サーバー・UI との通信，アルゴリズムの実行をそれぞれマイクロサービスとして分割している．プログラム全体をステートマシンとして構築して，サービス全体で状態を共有できる仕組みにしている．補助的なものとして，一方向に送受信可能なデータ

チャネルを用いて相互に通信を行うようにしている．

実行時は，最適化された単独バイナリとして実行できるようにビルドしている．

3.2 UI

React を利用し，サーバーからのイベントストリームを受け取り，ソルバーの進捗をリアルタイムで表示するようにしている．ソルバーと完全に分離することで，時間のかかる処理が行われている間も UI はブロックされない．