SuperCon2022予選　解法のアルゴリズム

チーム　T0M0K4ZU

1.愚直解

動的計画法＋経路復元で解く。

dpテーブル `tmap[SC_Nt][SC_L][SC_L]`を用意しておき、無限大(SC_Nt以上の値)で初期化しておく。

`tmap[t][i][j]` は、時刻tにおいてマス (i, j) に到達するのにかかる最小の誤差を表す。

はじめ、`tmap[0][(SC_L-1)/2][(SC_L-1)/2]`を0にして、無限大でない値のマスから上下左右に配るdpをする。最終的に最も値の小さいマスを最終地点とする経路が最短であることが分かる。

ここで、各マスへ移動するときに元のマスを保存しておいて逆順に辿ることで経路を復元できる。

計算量は $O(SC_{Nt} \times SC_{N})$ より、十分高速に厳密解が求まる。

平均的な速度は約1120[μs]であった。

2.実行と関係ない変数を作らない

明らかに無駄なコードを消した。1050[μs]程度になった。

3. SC_distance関数を毎回呼ばない

色は10色しかないので、毎回その差を計算するより100通りすべてのパターンを調べたテーブルを作って参照した方が速くなる。640[μs]程度になった。

4. 定数化

上下左右に移動するとき、周期境界条件で次のマスの (i, j) を計算するために、幅で割った余りを用いるが、これは定数テーブルにできるので毎回計算しなくてよい。550[μs]程度になった。

5. 確率的な枝刈り

dpして、最終的なコスト関数の値はほとんど(実験した中ではすべて)の場合で20未満になる事が分かった。

そこで、dp中に20以上の値が出ていたら、それ以降を枝刈りするようにした。510[μs]程度になったが、低確率で厳密解ではない答えを出力してしまう可能性がある。安全を取って23以上で枝刈りすることにした。530[μs]程度になった。

6. 変数の定義を何度もしない

2重以上のfor文の中で変数の宣言をしないようにした。for文よりも前に宣言しておいて、使いまわすようにした。

7. 最小値のif文で2度計算しない

上のような処理を、下のようにした。

```
if(min > calc(p)){
    min = calc(p);
    //処理
}
```

```
int q;
if(min > (q = calc(p))){
    min = q;  //二度計算しない
    //処理
}
```

この工夫で460[μs]程度になったが、-O2オプション下においてはそこまで高速にならなかった。

8. 2重for文を1重にする

上下左右に動くのを、(i, j)のまま二次元の配列で保存していたので、アクセスするたびにi*L + jを計算する必要があった。これを、そもそも(i*L + j)の一次元配列、つまりSC_Cと同じように保存した。アクセスするには、あらかじめ作っておいた上下左右に移動した後の番号を格納してある配列を見ればよいので、無駄な計算が無くなった。350[μs]程度になった。-O2下では、170[μs]程度だった。