2022 年 6 月 1 日

スーパーコン 22 予選・認定問題: ロボット追跡問題

1　ロボット追跡問題

様々な色のタイルを敷き詰めた床の上を移動するロボットがある. ロボットは床の上を動き回りながら各時刻に自分のいるタイルの色を送信し, 我々はそれを受信している. 我々は部屋の全てのタイルの色を知っている. さて, 我々の仕事は受信したロボットの報告から, 各時刻にロボットのいる位置を推定することである. このロボットには「ランダムウォーク」というテスト用のモードがあり, それで動作させたロボットを追跡するプログラムを考えよう.

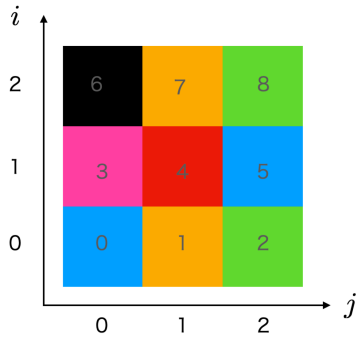

図 1　$L = 3$ の例. 中央のタイル $(1, 1)$ の上にいるロボットは上下左右に動ける. タイルの上の数字は $I = L \times i + j$.

床には図 1 のように正方形のタイルが L 行 L 列, 合計 $N = L \times L$ 枚敷き詰められている (L は奇数). 縦方向下から i 行目, 横方向左から j 列目のタイルの場所を (i, j) と表そう. ここで i,j ともに $0, 1, 2, \ldots, L-1$ の値をとりうる. タイルにはラベル $I = 0, 1, 2, \ldots, N-1$ も振られている. このラベルは $L \times i + j$ のように決まっている. タイルの色は N_{q} 種類で, $0, 1, \ldots, N_{\mathrm{q}} - 1$ という値（色番号）で区別する. I 番目のタイルの色番号 $C[I]$ はあらかじめランダムに選ばれていて, 我々は全てのタイルの色番号 $C[I]$ $(I = 0, 1, 2, \ldots, N-1)$ を知っている.

ロボットは最初, 時刻 $t = 0$ に部屋の中央 $(i, j) = ((L-1)/2, (L-1)/2)$ のタイルにいる. その後, ランダムに動いてゆく. ロボットは, 各時刻 t に必ずどこかのかタイル (i, j) にいて, 次の時刻 $t + 1$ には, 上下左右のいずれかの隣接するタイル, つまり

$(i+1,j)$,$(i-1,j)$,$(i,j-1)$,$(i,j+1)$ のどれかをランダムに選んで移動する. (したがって偶数ステップ後に元の場所に戻ってくることもある.) ロボットは各時刻 $t=1,2,\ldots,N_{\mathrm{t}}$ に自分がいるタイルの色番号 $C_{\mathrm{r}}[t]$ を送信する.

ロボットの各時刻での位置を $I^*[t]$ $(1 \leq t \leq N_{\mathrm{t}})$ とすると, ロボットの報告が正確ならば $C_{\mathrm{r}}[t] = C[I^*[t]]$ である（3,2 級認定問題). ロボットの報告にノイズが入って色番号がズレてしまう場合も考えよう（1 級認定問題兼予選問題).

問題ではタイルの色 $C[I]$ $(I=0,1,2,\ldots,N-1)$, ロボットの歩数 N_t, ロボットの報告 $C_{\mathrm{r}}[t]$ $(t=1,2,\ldots,N_{\mathrm{t}})$ が与えられる. 出力して欲しいのは, あなたが推定したロボットの軌跡 $I[t]$ $(t=1,2,\ldots,N_{\mathrm{t}})$ である.

色の数 N_{q} が十分大きいと, 一つのタイルからみて, 上下左右の隣接しているタイルの色が全て異なっている場合がほとんどであろう. そうすると推定は易しそうだ.　しかし, 少なくなると...

境界条件

さて, ロボットが床の端まで来たらどうするか説明していなかった. 今回は簡単のために, 図 2 のような「周期境界条件」を考える. $(L-1,j)$ の上隣は (L,j) だが, そこは $(0,j)$ と同じとみなす. $(0,j)$ の下隣は $(-1,j)$ だが, そこは $(L-1,j)$ と同じとみなす. 同じように $(i,L-1)$ の右隣は $(i,0)$,$(i,0)$ の左隣は $(i,L-1)$ とみなす. つまり,$L \times L$ の床が無限に繰り返していると考える. これでどの場所にいても, 動ける方向は 4 つあることになる.

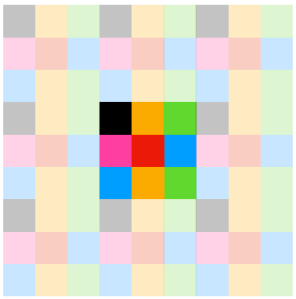

図 2　周期境界条件の概念図. 図 1 のパターンを繰り返している.

軌跡に対する条件

当然ながら, 経路 $I(t)(t=1,2,\ldots,N_{\mathrm{t}})$, は, 実際にありうるものでなければならない. すなわち $t=0$ に中心にいたロボットが一歩づつ動いて作れる経路でなくてはならず, 途

中でジャンプしたりしている経路は不適格とする.

入力

標準入力から以下の形式の入力ファイルを読み込む. このための関数はあとで説明するように提供するヘッダーファイルに含まれる.

$C[0]$
$C[1]$
$\vdots$
$C[N-1]$
$C_{\mathrm{r}}[1]$
$C_{\mathrm{r}}[2]$
$\vdots$
$C_{\mathrm{r}}[N_{\mathrm{t}}]$

出力

標準出力に以下の形式のファイルを出力する. このための関数もあとで説明するように提供されるヘッダーファイルに含まれる.

$I[1]$
$I[2]$
$\vdots$
$I[N_{\mathrm{t}}]$
入力読み込み終了からの経過時間 (秒)

2 問題

2.1 SuperCon 3 級認定問題

タイルの色 $C[I](I = 0, 1, 2, \ldots, N-1)$ とロボットの報告 $C_{\mathrm{r}}[t]$ $(t = 1, 2, \ldots, N_{\mathrm{t}})$ が問題として与えられる. ロボットの報告は正確で, ノイズはない. また色の数 N_{q} が十分多く, 全てのタイルについて, 上下左右の隣接するタイルの色は必ず全て異なっている. そうなっていないものは問題から除外されている. 各時刻におけるロボットの位置 $I[t]$ $(t = 1, 2, \ldots, N_{\mathrm{t}})$ を求めるプログラムを作れ. 各時刻でのタイルの色 $C[I[t]]$ が, ロボットの報告した $C_{\mathrm{r}}[t]$ に一致していれば正解とする. 1 問を解くプログラムを作れば良い.

2.1.1 3 級認定問題詳細

ヘッダーファイル sc3.h に必要な関数と配列が定義されているので,

```
#include "sc3.h"
```

として使うこと. 定義されているものは以下である.

`void SC_input()` : **問題を標準入力から読み込む関数**
`void SC_output()` : **解答を標準出力に出力する関数**
`# define SC_L  11` : **床の一辺の長さ**
`# define SC_N  121` : **タイルの総数**
`# define SC_Nq 100` : **タイルの色の数**
`# define SC_Nt 100` : **ロボットの歩数**
`int SC_C[SC_N]` : **問題のタイルの色を格納する一次元整数配列**
`int SC_Cr[SC_Nt+1]` : **問題のロボットの報告する色を格納する一次元整数配列**
`int SC_ans[SC_Nt+1]` : **解答を格納する一次元整数配列**

2.1.2 3 級認定基準

問題は 10 問与えられ,1 問を解くプログラムを問題を変えて 10 回実行する. 制限時間内にすべてに正解した場合に 3 級を認定する. 制限時間は入出力を除く計算時間で 10 問

合計 1 分とする.

2.2 SuperCon 2 級認定問題

タイルの色 $C[I](I = 0, 1, 2, \ldots, N-1)$ とロボットの報告 $C_{\rm r}[t]$ $(t = 1, 2, \ldots, N_{\rm t})$ が問題として与えられる. 3 級問題と同様にロボットの報告は正確で, ノイズはない. しかし, 上下左右の隣接するタイルの中に同じ色のものがある場合が除外されていない. 各時刻におけるロボットの位置 $I[t]$ $(t = 1, 2, \ldots, N_{\rm t})$ を求めるプログラムを作れ. 各時刻でのタイルの色 $C[I[t]]$ が, ロボットの報告した $C_{\rm r}[t]$ に一致していれば正解とする. 1 問を解くプログラムを作れば良い.

2.2.1 2 級認定問題詳細

ヘッダーファイル sc2.h に必要な関数と配列が定義されているので,

```
#include "sc2.h"
```

として使うこと. 定義されているものは以下である.

`void SC_input()` : **問題を標準入力から読み込む関数**
`void SC_output()` : **解答を標準出力に出力する関数**
`# define SC_L  11` : **床の一辺の長さ**
`# define SC_N  121` : **タイルの総数**
`# define SC_Nq 10`: **タイルの色の数**
`# define SC_Nt 100` : **ロボットの歩数**
`int SC_C[SC_N]` : **問題のタイルの色を格納する一次元整数配列**
`int SC_Cr[SC_Nt+1]` : **問題のロボットの報告する色を格納する一次元整数配列**
`int SC_ans[SC_Nt+1]` : **解答を格納する一次元整数配列**

2.2.2 2 級認定基準

問題は 10 問与えられ,1 問を解くプログラムを問題を変えて 10 回実行する. 制限時間内にすべてに正解した場合に 2 級を認定する. 制限時間は入出力を除く計算時間で 10 問合計 1 分とする.

2.3 SuperCon 1 級認定問題兼予選問題

タイルの色 $C[I](I = 0, 1, 2, \ldots, N-1)$ とロボットの報告 $C_{\mathrm{r}}[t]$ $(t = 1, 2, \ldots, N_{\mathrm{t}})$ が問題として与えられる.2 級問題と同じく, 上下左右の隣接するタイルの中に同じ色のものがある場合が除外されていない. さらにロボットの報告は不正確でノイズが含まれる. そこで, 以下に説明するコスト関数がなるべく小さくなるような各時刻におけるロボットの位置 $I[t]$ $(t = 1, 2, \ldots, N_{\mathrm{t}})$ を求めるプログラムを作れ.

2.3.1 1 級認定問題詳細

ノイズが入って色番号が真の値から最大 ± 1 ずれたものをロボットが報告する. ただし, 図 3 に示すように 0 番から -1 ずれた色は $N_q - 1$ 番,$N_q - 1$ 番から $+1$ ずれた色は 0 番とする. 詳しく述べると $C_{\mathrm{r}}[t] = C[I^*[t]] + \delta$ で, ズレ δ は -1,0,1 の値をそれぞれ確率

$$\frac{p_{\mathrm{noise}}}{2}, 1 - p_{\mathrm{noise}}, \frac{p_{\mathrm{noise}}}{2}$$

でとる. ここで $p_{\mathrm{noise}} = 0.1$ とする.

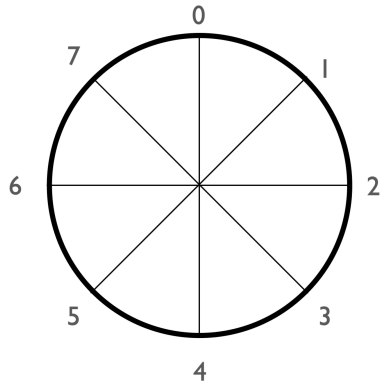

図 3　$N_q = 8$ の場合の例. $C = 0, 1, 2, \ldots, 7$ を時計回りに円周上に割り付ける. 0 の左隣は -1 ではなくて 7. 2 つの色番号の距離もこの円周上で定義する. 例えば 0 と 7 の距離は 1.

そこで次のようなコスト関数 (付録を参照) を定義し, これがなるべく小さくなる軌道

$I(t)(1 \leq t < N_{\mathrm{t}})$ を推定することにしよう.

$$E = \sum_{t=1}^{N_{\mathrm{t}}} \mathrm{SC\_distance}(C_{\mathrm{r}}[t] - C[I(t)]) \tag{1}$$

ただし SC_distance$(C_1 - C_2)$ は 2 つの色番号 C_1 と C_2 の差を調べる関数で, 図 3 のように円周上に色番号を並べたときの円周に沿って測った C_1 と C_2 の距離である. この関数はヘッダーファイルの中で与えられる.

ヘッダーファイル sc1.h に必要な関数と配列が定義されているので,

```
#include "sc1.h"
```

として使うこと. 定義されているものは以下である.

`void SC_input()` : **問題を標準入力から読み込む関数**
`void SC_output()` : **解答を標準出力に出力する関数**
`# define SC_L  11` : **床の一辺の長さ**
`# define SC_N  121` : **タイルの総数**
`# define SC_Nq 10`: **タイルの色の数**
`# define SC_Nt 100`: **ロボットの歩数**
`int SC_C[SC_N]` : **問題のタイルの色を格納する一次元整数配列**
`int SC_Cr[SC_Nt+1]` : **問題のロボットの報告する色を格納する一次元整数配列**
`int SC_ans[SC_Nt+1]` : **解答を格納する一次元整数配列**
`int SC_distance(int c)`: **色番号** `C1` **と** `C2` **の差** `c=C1-C2` **を入力するとその距離を返す関数**

2.3.2 1 級認定基準

問題は 10 問与えられる. 制限時間内にすべての問題について結果が出力され, 「50 ステップまででのコスト関数の値が 30 以下」である場合に 1 級を認定する. 制限時間は入出力を除いて 10 問合計 1 分とする.

2.3.3 本選出場チーム選考基準

問題は 10 問与えられる. 制限時間内にすべての問題について結果を出力することが必要である. コスト関数 (最終ステップまで含めた値) の合計でランキングし, 上位 20 チームを本戦出場チームとする. 制限時間は入出力を除いて 10 問合計 1 分とする. 以上の基

準でもし 20 チーム決まらなければ, 問題数を増やして（制限時間もそれに比例して増やす) 順位を決定する. またコスト関数で差がないチーム同士は, 計算時間の合計でランキングする. 成績がほとんど変わらない時には, レポートを参考に, アルゴリズム上の工夫が見られたチームを優先する.

3 言語と実行環境その他の注意

- OS: macOS Monterey ,CPU: プロセッサ 2.8 GHz Intel Core i5, メモリ 8GB を搭載したコンピュータ上でおこなう.
- コンパイラは Homebrew GCC 11.3.0_1 を使用し, 以下のように-O2 オプションでコンパイルする.
  g++-11 -O2 sc3.cpp
- 出力に関する注意:実行時に問題は標準入力から読まれ, 解答は標準出力に出力される. ヘッダーファイルに定義された入力, 出力用の関数を用いること. すべての級で,SC_input 関数は必ずプログラムの最初の実行文であること. つまり,SC _input の前に実行文を書いてはならない. 入出力は SC_input と SC _output のみで行ない, 他の出力をプログラムに含めてはならない
- 計算時間は time.h に含まれる clock() 関数で測定する. 入力終了後に測定を開始し, 出力直前までの所要時間を測る. また, 開始時刻と終了時刻を表す変数 SC_starttime と SC_ endtime の値はプログラム中で参照してよいが, 変えてはならない. なお, ヘッダーファイルは改変してはならない. これらの注意に反するプログラムは失格とする.

4 提出方法

ひとつの級にだけ応募してもよいし, 複数の級に応募してもよい. 言語は C++ を使用する. 作成したプログラムはファイル名を 1,2,3 級それぞれ,sc1.cpp,sc2.cpp,sc3.cpp として, 以下のファイルと一緒に zip ファイルにまとめて Google フォーム「SuperCon2022 予選・級認定申込」(https://forms.gle/vr1FLMEsCgzsKxVRA) から提出する. zip ファイルの名前は”〈チーム名〉.zip”にすること. 整理の都合上, 予選に応募しない場合も (ひとりでも) チーム名はつけること. なお, チーム名は 1 文字目が英字の 8 文字以内の英数字とする.

zip ファイルにまとめて提出するものは以下の 3 つである.

- 提出プログラム: sc1.cpp,sc2.cpp,sc3.cpp のうち応募する級に対応するもの. 予選応募には sc1.cpp が必須である.
- プログラム実行環境の説明（下で説明する入力生成器で作られたひとつの入力に対する実行結果と計算時間、使用コンパイラなど）”〈チーム名〉-env.txt”
- アルゴリズムのポイントや工夫などをまとめたレポート: ファイル名はテキストファイルの場合は”〈チーム名〉-algo.txt”,word ファイルの場合は”〈チーム名〉-algo.docx”,pdf ファイルの場合は”〈チーム名〉-algo.pdf”

級認定のみの応募はひとりでも構わないが,SuperCon 予選に応募する場合には 2 ないし 3 名のチームを作ること (予選通過後のチーム構成変更は認められないので注意). 1 級応募者は何も書かれていなければ予選参加とみなすので, 級認定のみ希望の場合はその旨上記 Google フォームで明記すること. 提出および問い合わせのメールアドレスは SuperCon ウェブサイトを参照のこと.

5 注意事項

問題解法を他のチーム (学校が同じか違うかにかかわらず) と相談してはならない. これは紳士協定だが, 厳守を期待する. また, 募集締め切りまでは, 問題解法に関して SNS やブログ等, チームメンバー以外の目に触れる場所に投稿することを禁止する. 解法に関する内容を含まない「SuperCon 予選問題の解きかたを議論しています」などの投稿は自由である. 募集締め切り後には解法についても SNS やブログに投稿して構わない.

付録 A 問題の入力生成器

C で書かれたプログラムを SuperCon ウェブサイトにて公開するので, 練習問題の生成に使ってよい. 実際の審査でも同じプログラムを使って問題を生成する. 例えば 3 級認定用の問題生成プログラムは sc3.zip に入っているのでこれをダウンロードして解凍して欲しい. 問題生成プログラムは gene_sc3.c, これに必要なヘッダーファイルは sc3_problem.h である. compile_gene_sc3.sh は問題生成プログラムをコンパイルするためのシェルスクリプトである. gene_sc3.sh は問題を生成するシェルスクリプトである. その中にある seed の値 (乱数を初期化するための整数) を変えると違う問題が生成できる. また sc3.h も入っているがこれは皆さんが作成するプログラムで使うものである. 2 級認定用,1 級認

定兼予選用も同様である.

なお, 問題生成のために必要な乱数にはメルセンヌツイスター法を用いている. メルセンヌツイスター法のライブラリ http://www.math.sci.hiroshima-u.ac.jp/m-mat/MT/SFMT/SFMT-src-1.5.1.zip をインストールして用いよ.

付録 B　色の距離関数

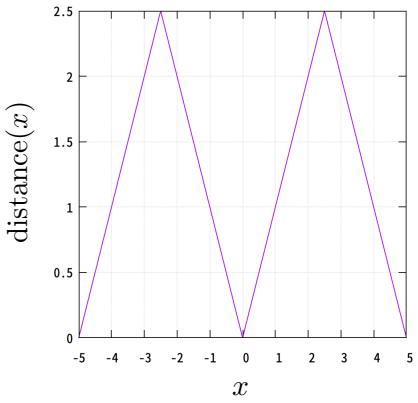

図 4　色の距離関数 SC_distance(x). $N_q = 5$ の場合の例. x は $-5, -4, \ldots, 4, 5$ の整数.

```
int SC_distance(int c){
  int a,b;
  a=abs(c);
  b=Nq-a;
  return (a > b) ? b : a;
};
```