本 選 課 題 解 説

スーパーコン２０１１審査委員会
大阪大学・東京工業大学

背景

「テトリス」や「ぷよぷよ」などの，いわゆる「落ち物ゲーム」を知らない人はいないだろう．テトリスやぷよぷよは,「計算理論」という数学・情報科学の研究分野においても興味深い題材である．それらは，いずれも計算の複雑さや困難さを表すクラスのうち「NP 困難」に属するため，一般にはコンピュータで解くことが難しい問題とされている．このような「ゲーム」の研究を通じて，そのゲームに類似の，応用上重要な別の問題を解くための足がかりが得られる場合が少なくない．そこで，今年の本選では，スーパーコン２０１１審査委員会が独自に考案した落ち物ゲームを解くプログラムを作成してもらう．このゲームは，テトリスを世に送り出した偉大なロシアに敬意を表して,「なくろん（наклон)」と命名された．

課題

「なくろん」

図のように，正方形の箱の中に，赤，黄，緑，青の玉（●，●，●，●,）及び壁（■）が配置されている（これを「盤面」と呼ぶ）．箱の周囲は，玉と同じ色の四角い穴（■，■，■，■）および壁（■）からなる「枠」で囲まれている．この箱を図の上下左右のいずれかの方向に傾けると，傾けた方向に玉が転がり，穴から落ちる．図１の状態から右に傾けた後の状態が図２である．同じ色の穴から落ちた玉の数が得点となる．与えられた盤面と枠（玉，壁，穴）に対して，より高得点となる箱の傾け方の順序（上，下，左，右）を求めるプログラムを作成してほしい．審査では巨大な盤面と枠を３問出題する．それぞれに対して制限時間内に最も高得点を出したチームが優勝である．

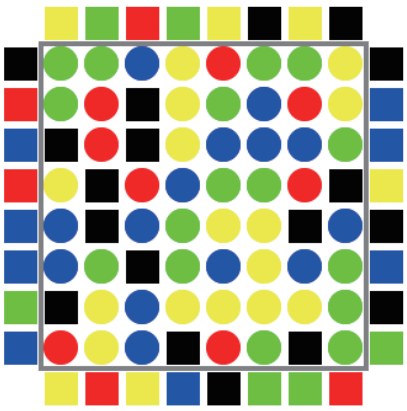

図１

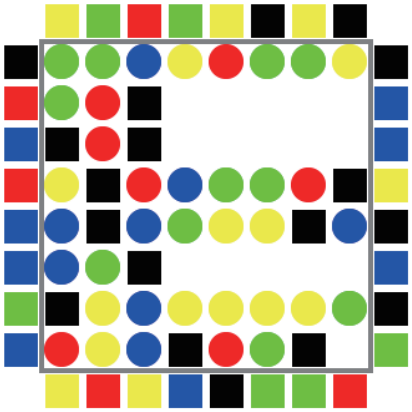

図２

課題の詳細

とりあえず，配布されたゲームで実際に遊んでみてほしい．そうすれば，このゲームのルールがすぐにわかるだろう．図１の状態から右に傾けることにより，壁の右側にある玉のうち，右の枠が壁であるもの以外の全てが穴から落ちて，図２の状態になる．この「傾ける」動作を,「なくろん」における「手」と呼ぶ．同じ色の穴から落ちた玉の数がその手の得点である．異なる色の穴から落ちた玉は得点にも減点にもならない．図１と図２の例の場合,「１手目」の得点は７点である．このような「手」を繰り返して，できるだけ多くの玉を同じ色の穴から落とすようにする．穴から同じ色の玉ばかりが落ちるわけではないから，ほとんどの場合に無駄が生じることに注意しよう．

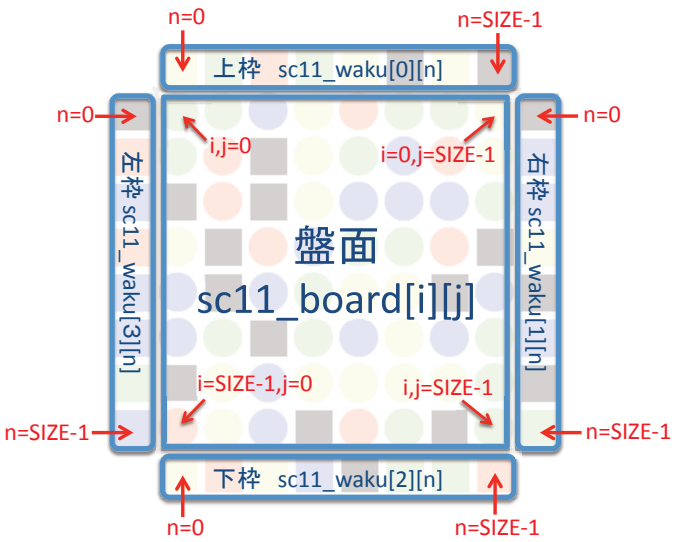

図３：盤面と枠の配列の定義

入力データは，配布するサンプルプログラム sc11_sample.c 内の入力関数 int sc11_init() によって，盤面（玉，壁）を表現する SIZE×SIZE の二次元配列 int sc11_board[i][j] (i,j=0, ..., SIZE-1) および枠（穴，壁）を表現する 4×SIZE の二次元配列 int sc11_waku[m][n] (m=0,1,2,3, n=0, ...,SIZE-1) にセットされる．図の例では SIZE = 8 である．配列 sc11_board[i][j] および sc11_waku[m][n] には，色番号（赤=1, 黄=2, 緑=3, 青=4），壁を表す番号 (5)，もしくは，空白を表す番号 (0) が入る．図３に示すように，盤面の配列の原点は左上とし，右方向に j が，下方向に i が増すものとする．例えば，図１の盤面においては，sc11_board[0][0]=3（緑玉），sc11_board[0][1]=3（緑玉），sc11_board[0][2]=4（青玉），sc11_board[1][0]=3（緑玉），sc11_board[1][1]=1（赤玉），sc11_board[1][2]=5（壁）であり，図２の盤面においては，sc11_board[1][3]=0（空白）である．枠 sc11_waku[m][n] においては，m=0 が「上枠」，m=1 が「右枠」，m=2 が「下枠」，m=3 が「左枠」を表す．上枠と下枠においては n は左から右へ増加することとし，右枠と左枠においては n は上から下へ増加することとする．図１の枠では，sc11_waku[0][0]=2（黄穴），sc11_waku[0][1]=3（緑穴），sc11_waku[1][0]=5（壁），sc11_waku[1][1]=4（青穴），sc11_waku[2][0]=2（黄穴），sc11_waku[2][1]=1（赤穴），sc11_waku[3][0]=5（壁），sc11_waku[3][1]=1（赤穴）である．

解は，配布されるヘッダーファイル（sc11.h）内で定義される１次元配列 int sc11_solution[k] (k=0,..., 29999) にセットすること．この３万手分の長さは念のため長めに用意してあるだけなので，全てを埋める必要はない．sc11_solution[k] には，宣言された直後には，値-1 が自動的にセットされる．各「手」において，箱を傾ける向きそれぞれに番号を与える．上＝０，右＝１，下＝２，左＝３とする．１手目に右に傾けるなら sc11_solution[0]=1，２手目に上に傾けるなら sc11_solution[1]=0 である．盤面によっては，どの方向に傾けても盤面が変わらない場合（例えば，１個の玉の前後左右が壁で囲まれている場合）や，何回か動かすと元の盤面に戻ってしまう

表 1: 審査に用いる問題の盤面と枠のパラメータ

	各色の玉の確率	盤面内の壁の確率	実行時間制限（分）
問題１	9/40	1/10	5
問題２	17/80	3/20	10
問題３	1/5	1/5	15

「周期状態」のような，いわゆる「千日手」状態に陥る場合があることに注意しよう．審査時には，sc11_solution[k] に 0,1,2,3 以外の値が入っているときは「停止（ゲーム終了）」とみなし，それ以前の手による得点の総和をその問題に対する「スコア」とする．仮にそれ以降に 0,1,2,3 があって得点の可能性があっても採点しない．

各チームが提出するプログラムは１つだけとする．プログラムのファイル名は"チーム名.c"とする．審査では，提出された各チームのプログラムに３個の問題を解かせる．３個とも盤面および枠の大きさは SIZE=250 である．あらかじめ配布する問題生成プログラム sc11_mkQ.cpp に対して，ある秘密の乱数の種を与えて生成する盤面および枠を審査に用いる．盤面を生成するためのパラメータは表１の値とする．各パラメータは確率であり，必ずしも異なる色の玉の数が厳密に同じになるわけではない．例えば，図１の盤面と枠は，表１の問題３と同じパラメータで生成されたものであり，各色の玉の個数の期待値は 8×8×1/5=12.8 個であるが，実際は，赤玉は８個，黄玉は１５個，緑玉は１６個，青玉は１４個であり互いに等しいわけではない．また，枠内の各色の穴と壁の確率は，全ての問題において等しく 1/5 とするが，同様に穴の数も色毎にばらつきがある．提出プログラムは各チームごとに１個だけだが，３個の問題はそれぞれ性質が異なり，有効なアルゴリズムが異なる可能性が高く，また，以下に示すように実行時間の制限も異なるので，プログラムの中でパラメータを分析することにより，３個の問題に対して異なる動作をするようにプログラムを設計することを勧める．

審査時には，表１に示すように，問題ごとに実行時間の制限を定める．各問題ごとの実行時間内に最後に出力された解を採点し，各チームのスコアを計算する．各問題ごとに各チームのスコアを比較し，１位になったチームには１１点，２位になったチームには１０点，...，１０位は２点，１１位は１点の「獲得点」を与える．全く出力がされない場合は０点とする．スコアが同点の場合は，より手の数の少ない方を優位として順位を決定する．手の数も同数の場合は，解が出力されるまでの計算時間がより短い方を優位とする．３問の中では，盤面内の壁の確率が高い方が，手の数が多くなり，アルゴリズムの工夫の余地も生まれ，計算時間も長くなることが予想されるので，問題２で１位になったチームにはボーナス点を１点．問題３で１位になったチームにはボーナス点を２点追加することとする．３つの問題の合計獲得点が最高点となったチームを優勝とする．上位２チームの合計獲得点が同点の場合は，３つの問題の総スコアが高い方を優勝とする．総スコアが同じ場合は，３つの問題の手の数の総和のより少ない方を優勝とする．手の数の総和も同じ場合は，３つの問題の計算時間の総和がより短い方を優勝とする．

プログラミングの注意

(1) プログラミング言語は C 言語．基本的に ANSI 準拠．GCC や Visual C++などに特有の仕様

は，スーパーコンピュータ（SX8R）では使えない可能性が高い．PC でコンパイルできたものが SX8R でコンパイルできるとは限らないことに注意．

(2) 並列化はしない．コンテスト期間中，各チームには SX8R の 1CPU（１コア）が割り当てられる．１チームが同時に実行できるプログラムは１つまでである．ただし，バッチジョブの投入自体は各チームごとに複数行ってもよい．その場合には１つのプログラムの実行が終了すると自動的に次のプログラムの実行が始まる．

(3) 配布されるベクトル化についての資料をよく読んで，プログラムのベクトル化率を上げるようにすること．ベクトル型スーパーコンピュータである SX8R は，ベクトル化率を上げることにより，1CPU でも PC の数百倍から数千倍のパフォーマンスを得ることが可能である．

(4) データの読み込みと解答の出力には審査委員会で用意したヘッダーファイルとサンプルプログラム `sc11_sample.c` 内の関数 `sc11_init()` および `sc11_output()` を使う．この関数は書き換えてはならず，入出力にこれ以外の関数を使ってはならない．

(5) 解答はひとつの解が出るたびに出力してよい．すなわち，解 `sc11_solution[k]` が得られた時点で出力用関数 `sc11_output()` 関数を呼んでよい．複数の解が出力された場合，最後のものを採点対象とするので，スコアを記録しておいて，より高いスコアを与える解が得られたときだけ，出力すること．

(6) 使用可能メモリーサイズは 31GB まで．ただし，ひとつの配列として取れるサイズに上限はない．

(7) その他 SX8R でのコンパイルやジョブの投入の方法の詳細，ベクトル化や並列化に関する注意などもコンテスト Wiki に掲載するので参照すること．

課題攻略のヒントと注意

配布する「問題生成器」（盤面と枠のデータ生成プログラム）(`sc11_mkQ.cpp`) を使って，いろいろな盤面と枠を生成して，プログラムに解かせてみよう．ただし，盤面のサイズ `SIZE` を最初から審査時と同じ 250 などの大きな値で試すのはやめておいた方がよいだろう．

審査に用いる問題のサイズ `SIZE=250` では，１０分程度の実行時間では，スーパーコンピュータを使っても全ての手を「枚挙」して「全探索」で最適解を得ることは不可能であることに注意しよう．よって,「枝刈り」や，探索の「打ち切り」などを工夫する必要がある．

最も高速だが高いスコアは望めそうにない解は,「１２３４１２３４１２３４...１２３４」などの「決め打ち」だろう．ただし，最悪１個も解が求まらない時の備えに，最初に一発このような「決め打ち」の解を出力しておいて損はないかも知れない．次に簡単だが，決め打ちと同様に高いスコアは望めそうにない解は,「ランダム」である．これはサンプルプログラム `sc11_sample.c` にも入れてある（`sc11_solver_random()`）ので試してみてほしい．

次に考えうる簡単な解は，各「手」において，３通り（最初の１手以降はひとつ前の手と同じ手は除くので）の「手」のうち，得点が最大になるものを選択する「欲張り法（greedy)」であろう．これもサンプルプログラム `sc11_sample.c` に入れてある (`sc11_solver_greedy()`）ので試してみてほしい（ただし，sc11_solver_greedy() では，わかりやすさを優先して，ひとつ前の手を除かず，４通りの手のうち得点が最大になるものを選ぶ仕様となっている）．

他にも２手先（９通り）までの得点の最大の手を選ぶという方法（2 ステップ-greedy）なども考えられる．各手で最適化する「評価関数」も色々と工夫してみてほしい．