1 万円ゲームについて

中野浩嗣
広島大学

1 はじめに

学生のプログラミング能力の向上を目的に，プログラミングコンテストなどがよく行なわれています．課題を理解するのに時間がかかったり，複雑なデータ構造を用いる必要があると，学生にとって参加する敷居が高くなります．ここでご紹介する 1 万円ゲームはプログラミングコンテスト用に考案した対戦型ゲームです．ルールは極めて単純で，C 言語プログラミングの学習を始めたばかりの学生でも参加することができます．ただし，単純なゲームのわりに奥が深いので，強いプログラムとするためには，さまざまなアルゴリズム的・プログラミング的な技を駆使する必要があります．たぶん，ゲーム理論で言うところの，二人零和有限確定不完全情報ゲームに分類されると思いますが，私は，ゲーム理論をさわりだけしか知らないので，実はよく知られたゲームなのかもしれません．私の研究室内で全員参加の大会を開催したところ，多くの学生が研究そっちのけで熱中し，大いに盛り上がりましたので，ご紹介いたします．

2 1万円ゲームとは

1 万円ゲームとは，二人の競技者 (プレイヤー A と B とする) が対戦するゲームです．各プレイヤーは最初に 1 万円ずつ持っています．この 1 万円を 10 回のラウンドに分けて，場に賭けます．そのときに，少ない金額を出した方が総取りです．同じ金額なら，それぞれが自分の賭金を取ります．つまり，各ラウンド i $(1 \leq i \leq 10)$ の A と B の賭金を，それぞれ a_i と b_i とすると，次のようになります．

- $\sum_{i=1}^{10} a_i = \sum_{i=1}^{10} b_i = 10000$.（賭金の合計は 10000 円）

- 各ラウンド i $(1 \leq i \leq 10)$ において， $a_i \geq 0$ かつ $b_i \geq 0$．（賭金は 0 円以上）
- $a_i < b_i$ のとき，A が $a_i + b_i$ 円を得る．$a_i > b_i$ のとき，B が $a_i + b_i$ 円を得る．$a_i = b_i$ のとき，A と B はそれぞれ a_i $(= b_i)$ 円を得る．（少ない方が勝ち）

少ない方が総取りというのがミソで，多い方が総取りだと，明らかにゲームとして成立しません．また，ラウンド数が 2 の場合も最強戦略は自明です．

プレイヤー A と B は，賭金 a_i と b_i を決定するときに，過去の賭金 $a_1, \ldots, a_{i-1}$ と $b_1, \ldots, b_{i-1}$ を知っていて，この情報を利用できるものとします．10 ラウンド目が終了後，A と B は合計で 20000 円持っていることになります．このときに，1 円でも多い方が，この対戦の勝者です．そして，この対戦を 1000 回連続で行ないます．この連続対戦中に過去の対戦履歴を利用することができ，相手の賭金の傾向を分析して戦略を決定することもできます．1000 回の対戦行なった結果，勝ち数の多い方が，１対１対戦の勝者となります．研究室内では，総当たりを行ないました．

プログラミングコンテスト課題は，この競技者の賭金を決めるプログラムを C 言語でつくることです．対戦管理プログラムが用意されており，対戦プログラムはプレイヤー A とプレイヤー B のプログラムを子プロセスとして起動します．賭金のやり取りは，パイプを用います．各対戦プログラムは，まず a_1 と b_1 を標準出力に書き込みます．対戦管理プログラムは，これらを受け取り，このラウンドの結果を表示します．そして，b_1 と a_1 をプレイヤー A と B のプログラムにパイプを通して送り，各プログラムはこの値を標準入力から読み込みます（図 1）．これを，同様に 10 ラウンド繰り返します．対戦回数は子プロセスを起動するときの引数で与えるものとし，その回数だけ対戦を繰り返します．

3　１万円ゲームのプレイヤープログラムの例

プレイヤープログラムの簡単な例を紹介します．

always1000.c: 10 ラウンドとも 1000 円を賭けるプログラム．

minus1.c: 1 ラウンド目は 0 円を賭けます．2〜9 ラウンドは相手の直前のラウンドの賭金から 1 円引いたものを賭けます．10 ラウンド目は残り全額を賭けます．

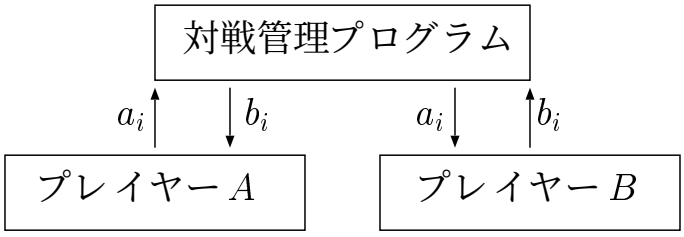

図 1: 対戦管理プログラムとプレイヤープログラム

random.c: 9 ラウンド目は残りの金額から 0 円までの一様乱数で賭金を決めます.

リスト 1, 2, 3 はこれらのプログラムリスト（ヘッダファイルは省略）です. 極めて簡単にプログラムを書くことができます.

リスト 1: always1000.c

```
int main(int argc,char **argv){
  int n,j,k,num_game;
  num_game=atoi(argv[1]);
  for(k=1;k<=num_game;++k)
    for(j=1;j<=10;++j){
      printf("1000\n");
      fflush(stdout);
      scanf("%d",&n);
    }
  exit(0);
}
```

では，プログラムを対戦させてみましょう. ここでは，簡単のため 1 回勝負とします. リスト 4 は，対戦管理プログラムの出力です. 対戦終了後，alwasy1000.c が 3008 円，minus1.c が 16992 円獲得し，minus1.c の勝ちになります. 直感的に，相手の賭金よりわずかに少ない金額を賭けるのがよさそうだということがわかります. random.c は乱数を使うので，勝ったり負けたりします. random.c は always1000.c に負け，minus1.c に勝つことが多いので，この 3 つのプログラムは三すくみになっています.

リスト 2: minus1.c

```
int main(int argc,char **argv){
  int n,j,k,num_game;
  num_game=atoi(argv[1]);
  for(k=1;k<=num_game;++k){
    int bet,sum=0;
    printf("0\n"); // round 1
    fflush(stdout);
    scanf("%d",&n);
    for(j=2;j<=9;++j){ //rounds 2-9
      bet=(n>0?n-1:0);
      printf("%d\n",bet);
      fflush(stdout);
      sum+=bet;
      scanf("%d",&n);
    }
    printf("%d\n",10000-sum); // round 10
    fflush(stdout);
    sum+=bet;
    scanf("%d",&n);
  }
  exit(0);
}
```

4 １万円ゲームのいろいろな戦略

賭金の合計は 10000 円なので，片方のプレイヤーが全てのラウンドを勝つことはできず，勝ったり負けたりします．相手が大きな金額を賭けたときに，それよりわずかに小さい金額を賭けると，その合計を獲得することができ，勝ちにつながります．1000 回の対戦の間に，相手の賭金の傾向や戦略を推定して対応していけば，強いプログラムになりそうです．この１万円ゲームプログラミングコンテストで学生が考えた戦略をいくつか紹介します．

- 1 ラウンド目に比較的大きな金額（例えば 7000 円ぐらい）をランダムに賭け，残りのラウンドは残金をほぼ均等に（ランダムに多少増減させ）賭ける．1 ラウンド目は負けるが，相手の賭金が平均の 1000 円程度であれば，負けは 8000 円程度である．残りの各ラウンドは，こちらは 300 円程度の賭金に対して，相手は 1000 円程度を出す必要がある．よって，残りの全ラウンドを勝つことができる可能性が高く，そうなれば，こちらは約 12000 円得ることができる．

リスト 3: random.c

```
int main(int argc,char **argv){
  int n,i,j,num_game;
  num_game=atoi(argv[1]);
  srand(time(NULL));
  for(j=1;j<=num_game;++j){
    int bet,remain=10000;
    for(i=1;i<=9;++i){
      bet=random()%remain;
      printf("%d\n",bet); // rounds 1-9
      fflush(stdout);
      remain-=bet;
      scanf("%d",&n);
    }
    printf("%d\n",remain); // round 10
    fflush(stdout);
    scanf("%d",&n);
  }
  exit(0);
}
```

- 直前の対戦で，相手が各ラウンドで賭けた金額を記憶しておき，1〜9 ラウンドでその金額より少ない金額を賭け，10 ラウンド目は残金を賭ける．相手が直前の対戦と同じ傾向なら，1〜9 ラウンドを勝つことができる．直前だけでなく，過去の複数回の対戦を記憶しておき，相手の賭金のラウンドごとの平均から同様のことを行なうことも考えられる．

- 上記を含めた，いろいろな戦略をランダムに選択する．単にランダムにするのではなく，各戦略の勝率を記録しておき，高い勝率の戦略ほど頻繁に選択されるようにする．

実際に学生が考えた戦略はさまざまであり，かなり複雑なものもありました．

ただし，全ての戦略に勝つという最強な戦略はなさそうです．ある戦略 A_1 に対して，それに勝つことができる戦略 A_2 を作ることは必ずできると思われます．一般に，戦略 A_i に対して勝つ戦略 A_{i+1} ができれば，どんどん強い戦略がつくれるように思われます．ところが，A_n は $A_1, A_2, \ldots A_{n-1}$ の全てに勝つことができるかというと，A_{n-1} には勝てても，それ以外の戦略には勝てる保証はありません．例えば，３つの具体的戦略 A，B，C が存在して，これら全てに勝つ戦略は存在しないといような類いのこと

リスト 4: always1 と minus1 の対戦結果

```
always1000 vs minus1
 1: 1000 [ 0] -> 0 1000
 2: 1000 [ 999] -> 0 2999
 3: 1000 [ 999] -> 0 4998
 4: 1000 [ 999] -> 0 6997
 5: 1000 [ 999] -> 0 8996
 6: 1000 [ 999] -> 0 10995
 7: 1000 [ 999] -> 0 12994
 8: 1000 [ 999] -> 0 14993
 9: 1000 [ 999] -> 0 16992
10: [1000] 2008 -> 3008 16992
     3008 [16992]
minus1 won.
```

が証明できるかもしれません．

5 おわりに

1万円ゲームは，単純な割りに奥が深い対戦型ゲームです．C言語プログラミングの入門者でも，参加することができます．対戦管理プログラムを含めたソースコードは，http://www.cs.hiroshima-u.ac.jp/~nakano/10000/にありますので，ご利用ください．