プログラミング言語実験
第4回
コンピュータ大貧民
— より高度なクライアント機能の実装
（場が空の時のジョーカーを利用したペアの提出） —

2019年05月06日、07日

2　場が空の時のジョーカーを利用したペアの提出

2.1　アイデア

今のプログラムでは、ジョーカーは単騎でしか出すことができません。そこで、今度はジョーカーを利用したペアを出せるようにしてみましょう。

今回は、次の手順を踏むことで、手持ちのカードからペアを発見し、提出することを考えます。

1. ジョーカーを利用してペアとして出せるカードの情報を持つ配列 info_j_table[8][15] を新しく作成する。
2. info_j_table[8][15] を探索し、一番弱いペアを発見する。
3. 一番弱いペアの情報を提出手として配列に入れる。

新しく作成する info_j_table について説明します。たとえば、手持ちのカードの配列が 次の図のとおりとします。

my_cards：

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0				1			1		1						
1		1		1			1								
2							1								
3															
4		2													
5															
6															
7															

ジョーカーを持っていますので、my_cards[4][1] に 2 が入っていることを確認しましょう。このとき、info_j_table として次のような配列を作成します。

info_j_table：

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0															
1															
2															
3															
4	1	2	1	3	1	1	4	1	2	1	1	1	1	1	1
5															
6															
7															

何をしているかというと、info_j_table の 4 行目に、その強さのカードが手持ちに何枚あるかを書いているだけです。ジョーカーはどのカードとしても出せるので、**どのマス目も、ジョーカーを利用せずに出せるペアの枚数+1** になっていることを確認しましょう。また、**ジョーカーを持っていないときは、配列のすべての値に 0** が入ることとします。

作成した後、ある枚数以上のペアや最大サイズのペアを発見したいなら、それらの組み合わせがあるかを 4 行目を見ることによって発見します。どの強さのカードをペアとして出すかが決まったら、今度は手持ちのカードの配列のその列を、提出する配列 select_cards にコピーすれば良いことになります。

次の小節から、この処理を行うプログラムを実際に書いていきます。

2.2 info_j_table の作成

前回、手札情報 my_cards から info_table を生成する関数 make_info_table を作成しました。今回は、この**関数を拡張する**ことで info_j_table も作成するようにしてみます。

前回の時点での、make_info_table のプロトタイプ宣言は以下のようになっています。

Listing 1: 前回の時点での make_info_table のプロトタイプ宣言

```
1 void make_info_table(int info_table[8][15], int my_cards[8][15]);
```

今回、info_j_table も作成することにしますので、引数が 1 つ増えます。ですので、プロトタイプ宣言は、次のように書き換えられます。

Listing 2: make_info_table のプロトタイプ宣言

```
1 void make_info_table(int info_table[8][15], int info_j_table[8][15],
2                      int my_cards[8][15]);
```

第 2 引数として info_j_table[8][15] が増えています。

作業：プロトタイプ宣言の書き換え

daihinmin.h に書かれている make_info_table のプロトタイプ宣言を書き換えましょう。

続いて、実際に make_info_table が今回行う処理について考えましょう。先ほど見たとおり、info_j_table[4][i] に入る値は、その値のカードの枚数+1 です。その値のカードの枚数は、すでに計算する処理を make_info_table で行っており、その結果は info_table[4][i] に入っています。ですの

で、info_j_table[4][i] の値は、info_table[4][i] に 1 を加えたものが保存されていればよいことになります。式で書くと info_j_table[4][i]=info_table[4][i]+1 です。従って、今回は、この計算を i=1〜13 に対して行うプログラムを書き加えればよいことになります。

作業：関数 make_info_table の追加

daihinmin.c の make_info_table を修正しましょう。

作業：make_info_table の説明

make_info_table が、どのようにして必要な処理を行っているか、説明しなさい。

2.3 ペアの発見と提出カードの設定

info_j_table ができたので、ここからペアを発見し、提出するカードとして設定します。ペアとして出せる手を発見するためには、info_j_table の 4 行目を調べていき、2 以上の値が入っているところを探せばよいことになります。

今回は、ジョーカーを利用して一番弱いペアを発見する関数 search_low_pair_wj を作成します。この関数は、自分の手札と info_j_table から、ジョーカーを利用して提出できる一番弱いペアを発見し、その結果を dst_cards に設定します。加えて、ペアを発見できたときは、返り値として 1 を、見つからなかった場合は 0 を返します。たとえば、

my_cards：

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0				1			1		1						
1		1		1			1								
2							1								
3															
4															

と

info_j_table：

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0															
1															
2															
3															
4	1	2	1	3	1	1	4	1	2	1	1	1	1	1	1

から、

dst_cards：

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0		2													
1		1													
2															
3															
4															

を設定します（5 行目以降は本当はありますが、省略しています)。ジョーカーを利用するために、dst_cards に 2 が入っています。プロトタイプ宣言は次のようになります。

Listing 3: search_low_pair_wj のプロトタイプ宣言

```
1 int search_low_pair_wj(int dst_cards[8][15], int info_j_table[8][15],
2                        int my_cards[8][15]);
```

作業：search_low_pair_wj のプロトタイプ宣言の追加

search_low_pair_wj を daihinmin.h に新たに追加しましょう。

この処理を実現するためには、どうしたらよいでしょうか。今回は、次のような方針で行くことにします。まず、info_j_table の 4 行目を左から右に順番に見ていき、2 以上の値を探します。そして、見つかった列を my_cards から dst_cards にコピーします。コピーする際、1 カ所だけ、0 の部分を 2 にしておきます。

作業：search_low_pair_wj の追加

search_low_pair_wj を daihinmin.c に新たに追加しましょう。

作業：search_low_pair_wj の説明

search_low_pair_wj が、どのようにして必要な処理を行っているか、説明しなさい。

2.4 提出するカードとして設定する

info_j_table を作成し、そこから最も弱いペア情報を取り出すことまでできるようになりました。ただ関数を準備しただけですので、まだカードを出すことはできません。最後に、select_cards_free で、作成した関数を呼び出すことにより、場が空のときはジョーカーを利用した最も弱いペアを優先的に出すようにしましょう。今回は、ジョーカーを使わないペアが手持ちになかったときに、ジョーカーを利用したペアを提出するようにします。そのために、次のことを select_cards_free 内で行うように**追記**します。

1. ペア情報を保存する配列 info_j_table を宣言する。
2. 自分のカード情報 my_cards からペア情報を作成し、配列 info_j_table に保存する。
3. 自分のカード情報 my_cards とペア情報 info_j_table から、最弱のペアを発見し、selesct_cards に保存する。

作業：select_cards_free の修正

上記の内容を select_cards.c の select_cards_free に追記しましょう。

作業：select_cards_free の説明

追記した select_cards_free が、どのようにして必要な処理を行っているか、説明しなさい。

これで、場が空で手持ちに階段もペアもなかった場合、ジョーカーを利用した最弱のペアを提出するようになりました。info_j_table には、ジョーカーを利用したすべてのペアの候補が保存されていますので、この配列の探索方法を変更すれば、他のペアを優先することもできます。

余裕のある人向け：最大枚数ペアの提出

場が空の時、手持ちのカードから、ジョーカーを利用した最大枚数のペアを優先的に提出したい。どのようにすれば実現できるか考えなさい。