プログラミング言語実験
第4回
コンピュータ大貧民
— クライアントの基本機能の実装（階段出し機能の実装） —

2019年05月06日、07日

3　場が空の時の階段の提出

3.1　階段提出のアイデア

次に、場が空の時に、階段を提出することを考えましょう。今回も、ジョーカーは考えません。ここでは、次の手順を踏むことで、手持ちのカードから階段を発見し、提出することを考えます。

- 先ほどの配列 info_table に、そのカードを起点として出せる階段の最大枚数を保存する。
- info_table を探索し、一番弱いカードから始まる階段を発見する。
- 一番弱いカードから始まる階段の情報を提出手として配列に入れる。

「ペア出し機能の実装」で作成した info_table には、ペアの情報が保存されています。この配列 info_table に追記し、そのカードを起点として出せる階段の情報も保存することにします。新しく保存する内容は、そのカードを起点として作ることが出来る階段の最大枚数です。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		1	1	1											
2							1	1							
3															
4															

から、

info_table：

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

を作成します（5 行目以降は本当はありますが、省略しています）。info_table のうち、追記されたのは 0 行目から 3 行目までで、4 行目は前に作成したペア情報です。info_table[0][3] の 4 は、スペードの 5 からはじまる 4 枚の階段を作れることを表しています。同様に、info_table[0][4] の 3 は、スペードの 6 からはじまる 3 枚の階段を作れることを表しています。

最小カードから始まる階段を見つけたい場合は、左の列から 3 以上の値を持つ場所を探していき、見つかったところからの階段を提出すればよいことになります。

3.2 info_table の作成

手札情報 my_cards から info_table を生成する関数 make_info_table を書き換え、階段情報も生成するようにします。先ほど見た階段情報とは、結局、「そのカードから何枚連続して並んでいるか」ということです。そのカードを持っていなければ、当然 0 が入ります。そのカードを持っていたら、**そのカードよりも 1 つ強いカードから連続している枚数 +1** が、そのカードから始まる連続した枚数になります。たとえば、「3.1　階段提出のアイデア」で示した my_cards では、スペードの 5 からの連続した枚数は、スペードの 6 から連続している枚数 +1、つまり 4 になります。

作業：実装内容の検討

make_info_table の書き換え内容を考えてみましょう。

作業：関数 make_info_table への追記

make_info_table に書き換え内容を追記しましょう。

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。

3.3 階段の発見と提出カードの設定

info_table に階段の情報が保存できたので、ここから最も弱いカードから始まる階段を発見し、提出するカードとして設定します。階段として出せる手を発見するためには、info_table の 0〜3 行目を調べていき、3 よりも大きい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、一番弱いカードから始まる階段を発見する関数 search_low_sequence を作成します。この関数は、info_table から、一番弱いカードから始まる階段を発見し、その結果を dst_cards に設定します。加えて、階段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、

info_table：

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

から、

dst_cards：

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0															
1		1	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）。プロトタイプ宣言は次のようになります。

Listing 1: search_low_sequence のプロトタイプ宣言

```
1 int search_low_sequence(int dst_cards[8][15], int info_table[8][15]);
```

作業：search_low_sequence のプロトタイプ宣言の追加

search_low_sequence 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_table の各列の 0〜3 行目を左から右に順番に見ていき、3 以上の値を探します。そして、見つかった場所から右方向にその値の数だけ、dst_table に 1 を入れます。

作業：実装内容の検討

search_low_sequence の実装内容を考えてみましょう。

作業：search_low_sequence の追加

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

作業：search_low_sequence の説明

search_low_sequence 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。

3.4 提出するカードとして設定する

info_table に階段情報を記録し、そこから最も弱いカードから始まる階段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、階段を提出するようにしましょう。すでに、場が空のときは最も弱いペアを優先的に出すようにしていますが、さらに追記を行い、“一番弱いカードから始まる階段” → “一番弱いペア” → “一番弱い単騎” の優先度で提出するようにしましょう。全体の流れは次のようになります。

1. 階段とペア情報を保存する配列 info_table を宣言する。
2. 自分のカード情報 my_cards から階段とペアの情報を作成し、配列 info_table に保存する。
3. 階段とペアの情報 info_table から、最弱のカードから始まる階段を発見し、selesct_cards に保存する。

4. 階段がなかった場合、自分のカード情報 my_cards とペア情報 info_table から、最弱のペアを発見し、selesct_cards に保存する。

5. ペアも無かった場合は、最も弱いカードを単騎で出すように、selesct_cards に情報を保存する。

上の 1., 4., 5. は、すでに select_cards_free に書かれていますので、主に追記するのは 2. と 3. になります。

作業：実装内容の検討

select_cards_free の修正内容を考えてみましょう。

作業：select_cards_free の修正

追記内容を select_cards.c の select_cards_free に書き加えましょう。

作業：select_cards_free の説明

追記内容が、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、場が空のときには、最弱のカードから始まる階段も提出するようになりました。info_table には、すべての階段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、他の階段を優先することもできます。

余裕のある人向け：最大枚数の階段の提出

場が空の時、手持ちのカードから、最大枚数の階段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。