フォルダ JavaDemoPrograms にあるプログラムの実行法

フォルダ JavaDemoPrograms には拡張子が class の以下の 32 個のファイルがある。

1. ヒープに関する演習問題 4.3 の解答の実行ファイルは次の 8 つのファイルからなる
   Heap.class
   HeapDemo$1.class
   HeapDemo$2.class
   HeapDemo$3.class
   HeapDemo$4.class
   HeapDemo$5.class
   HeapDemo.class
   HeapPanel.class

2. リストに関する演習問題 5.4 の解答の実行ファイルは次の 6 つのファイルからなる
   List.class
   ListDemo$1.class
   ListDemo$2.class
   ListDemo$3.class
   ListDemo.class
   ListPanel.class

3. 二色木に関する演習問題 7.3 の解答の実行ファイルは次の 6 つのファイルからなる
   RBTDemo$1.class
   RBTDemo$2.class
   RBTDemo$3.class
   RBTDemo.class
   RedBlackTree.class
   RedBlackTreePanel.class

4. 凸包に関する演習問題 10.4 の解答の実行ファイルは次の 6 つのファイルからなる
   GrahamDemo$1.class
   GrahamDemo$2.class
   GrahamDemo$3.class
   GrahamDemo$WorkAlgorithm.class
   GrahamDemo.class
   GrahamPanel.class

5. 水平・垂直線分の交差列挙に関する演習問題 11.3 の解答の実行ファイルは次の 6 つのファイルからなる

HoriVertIntPanel.class
HVIntDemo$1.class
HVIntDemo$2.class
HVIntDemo$3.class
HVIntDemo$WorkAlgorithm.class
HVIntDemo.class

これらのファイルをまずダウンロードする。ここでは、以下の説明の都合上、ダウンロードされたファイルは e:¥JavaDemoPrograms にあるものとする（ドライブが異なるとき、たとえば、ドライブが d のときには、以下の説明で e:をすべて d:と置き換えればよい)。
さらに、Java のソフトは既にインストールされていて、Path も正しく設定されていて、java の実行ファイルを走らせることができる環境にあるものとする。
このとき、コマンドプロンプト環境のもと（コマンドを入力できる状態）での 1.のヒープの java プログラムの実行は、以下のようにしてできる。(なお、>はプロンプト。)

コマンドプロンプトの画面で
> e:
> cd e:¥JavaDemoPrograms
とする。すると、
e:¥JavaDemoPrograms>
と表示される。そこで、
java HeapDemo 1000 &（とタイプしてその後リターンキーを押す）
（java と HeapDemo の間には半角のスペース、HeapDemo と 1000&の間にも半角のスペースがある。大文字と小文字は区別してタイプすることが必要である。なお、ここでの 1000 はパネルの横幅が 1000 ドット分占めることを表す。したがって、1000 ドット分とれないときは 800 とか 900 とに変更することもできる。ただし、その分横幅が小さくなるので表示されない文字がでる。)

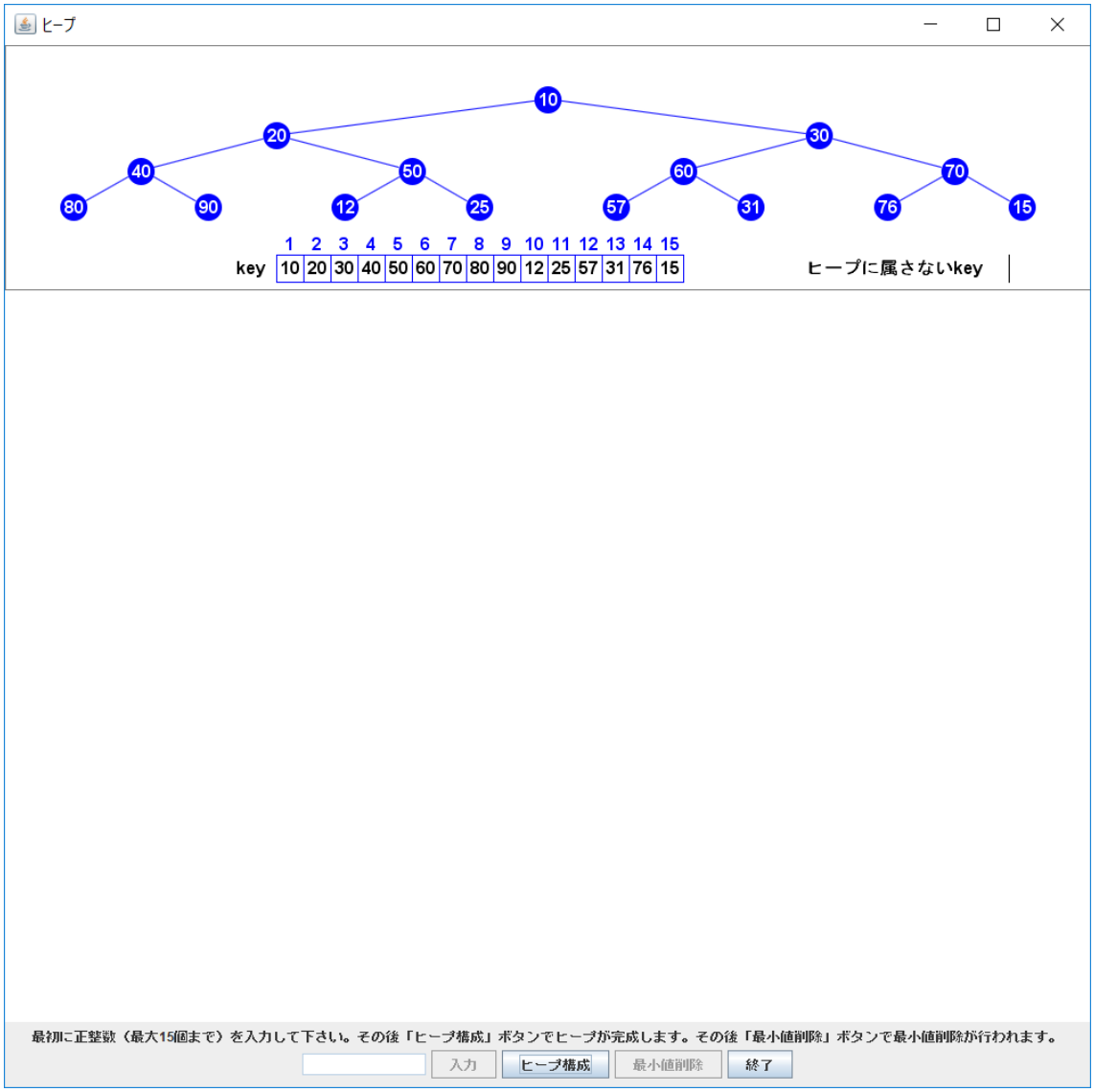
すると、1.のヒープのプログラムは実行される。具体的には、以下の図のようなパネルが開かれて、その画面の下の部分に

最初に正整数（最大 15 個まで）を入力して下さい。その後「ヒープ構成」ボタンでヒープが完成します。その後「最小値削除」ボタンで最小値削除が行われます。

と表示される。さらにその下に、白い長方形の箱があり、その右隣に「入力」「ヒープ構成」「最小値削除」「終了」のボタンがある。「入力」と「終了」のボタンが明るくなっていて、

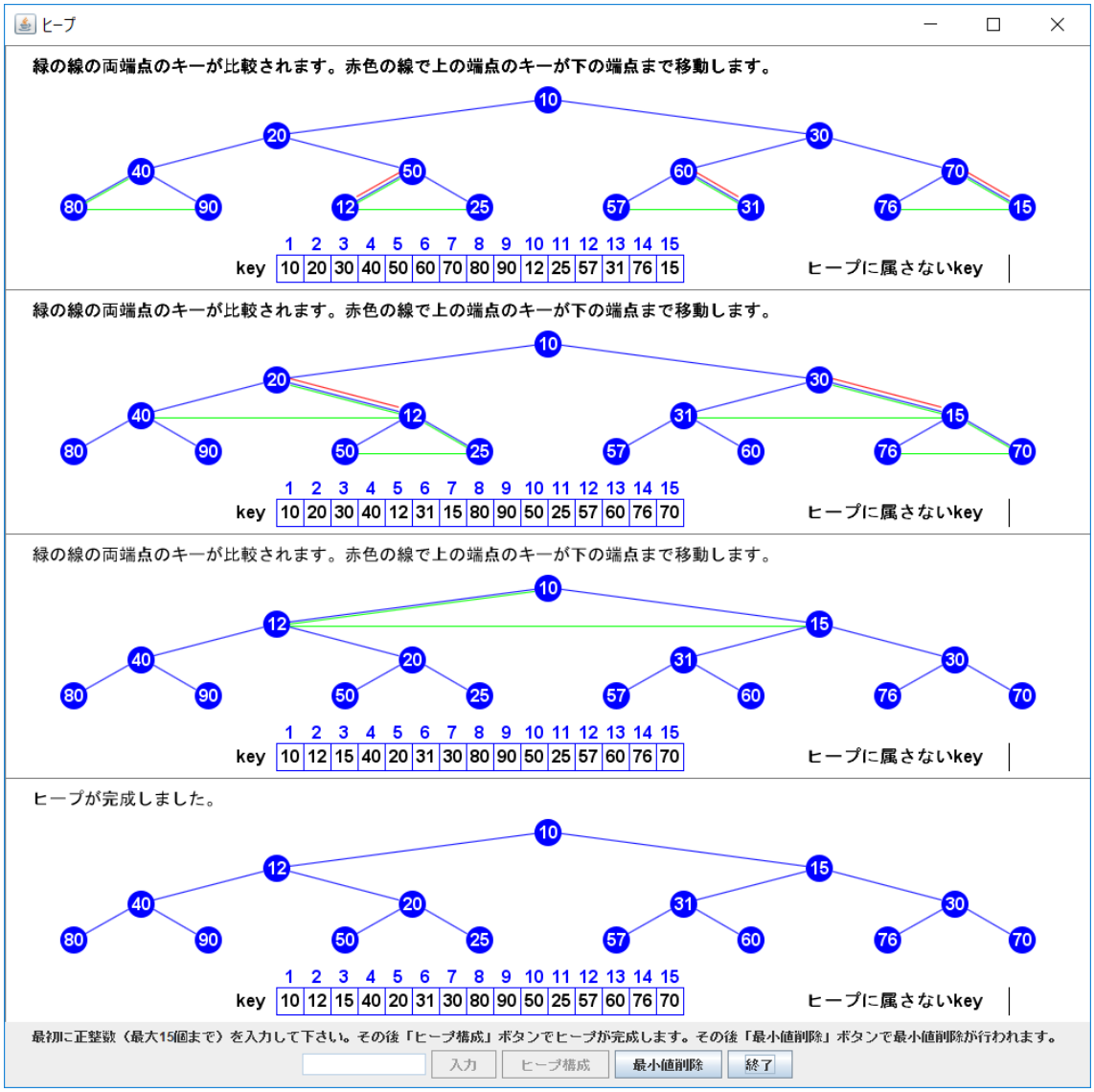
「ヒープ構成」「最小値削除」のボタンは薄く影が付けられている。

(a) そこで、白い箱の中にマウスを持って行って、2 桁の数字を半角で入力する。
　（たとえば、10 というようにタイプして「入力」のボタンを押す。
これを最大で 15 回繰り返す（最大で 15 個の 2 桁の数字が入力される。）
以下は、
　10, 20, 30,40,50,60,70,80,90,12,25,57,31,76,15
と 15 個入力した結果のパネルである。

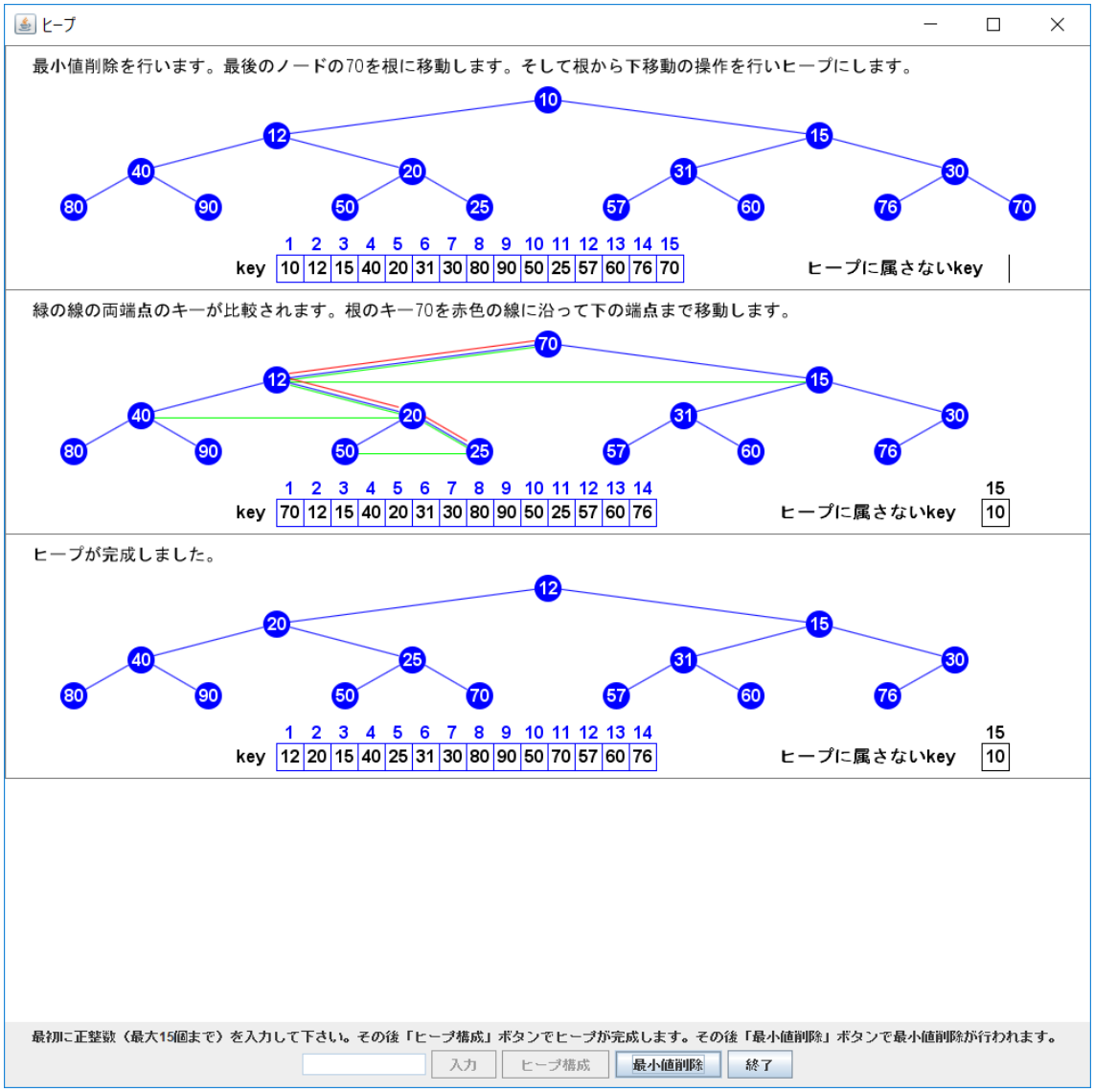


(b) 次に、「ヒープ構成」のボタンを押す。

（なお、うまくいかなかったときには、「終了」のボタンを押すと、強制終了され、はじめから繰り返すことができる。）するとパネルの内容は以下のようになる。

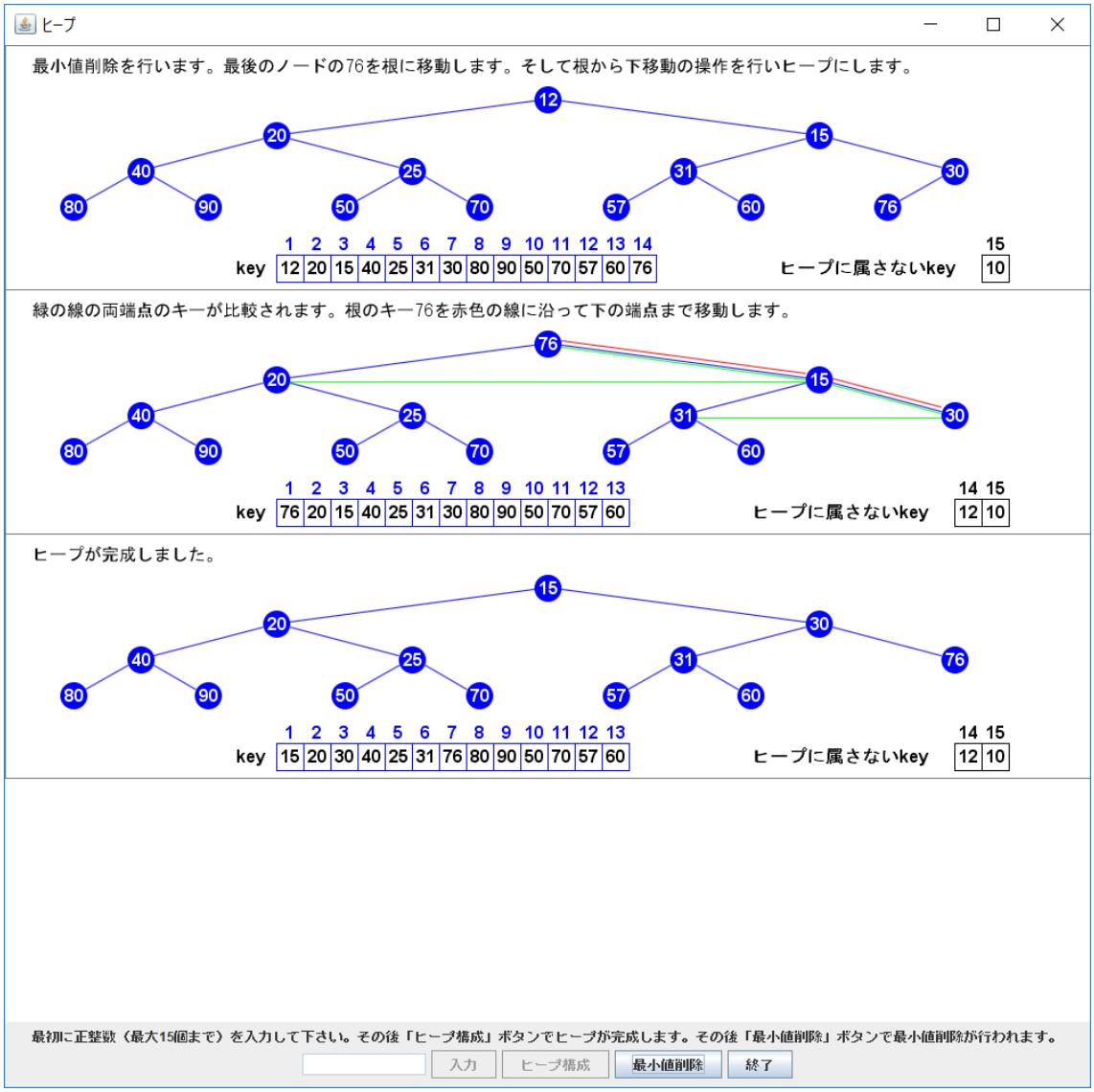


(c) それから、「最小値削除」のボタンを押すことを繰り返す。
（入力したデータの個数分繰り返す。）すると、パネルは以下のようになる。



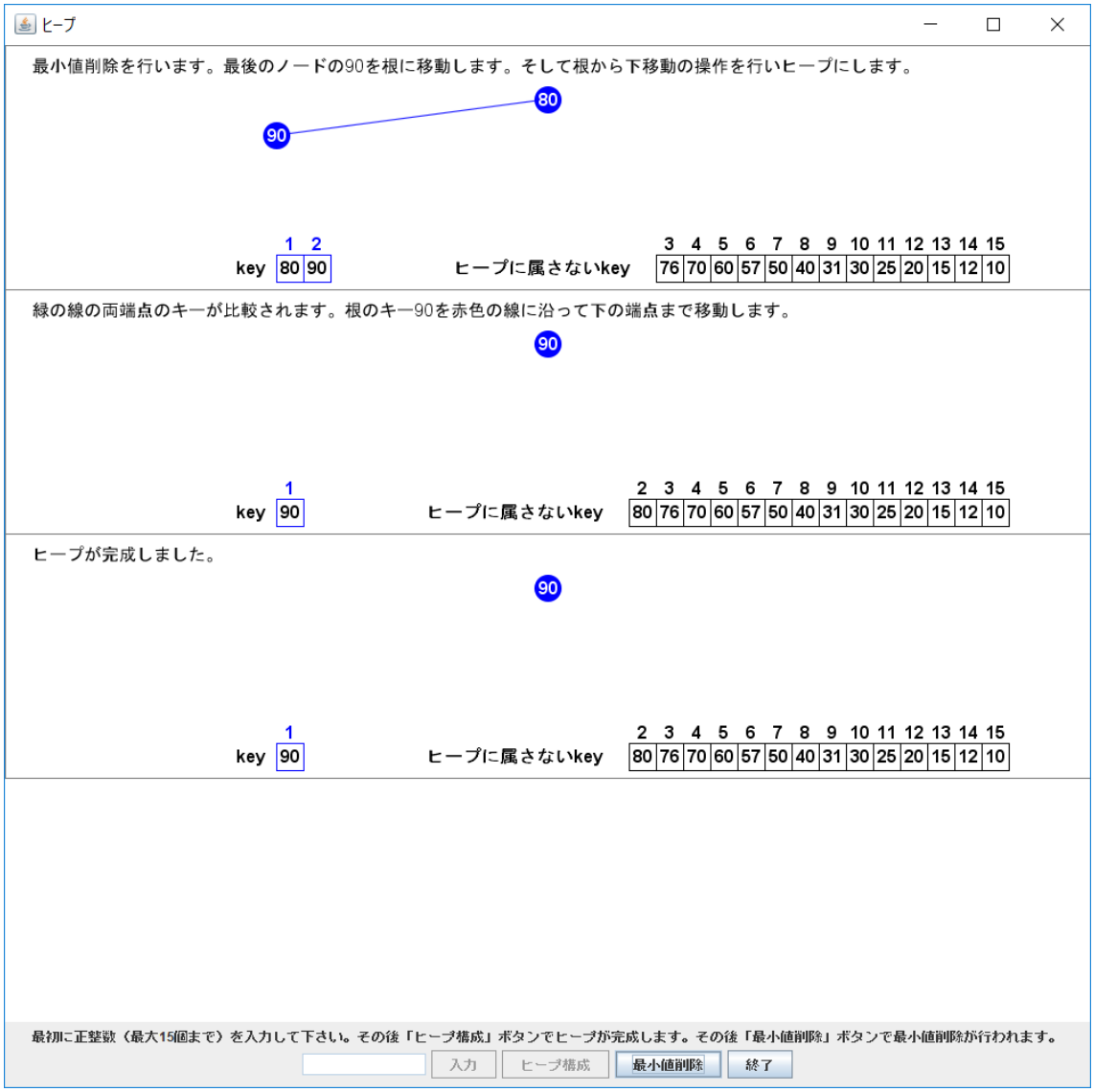
（最小値の 10 が削除された）

さらに「最小値削除」のボタンを押すとパネルは以下のようになる。



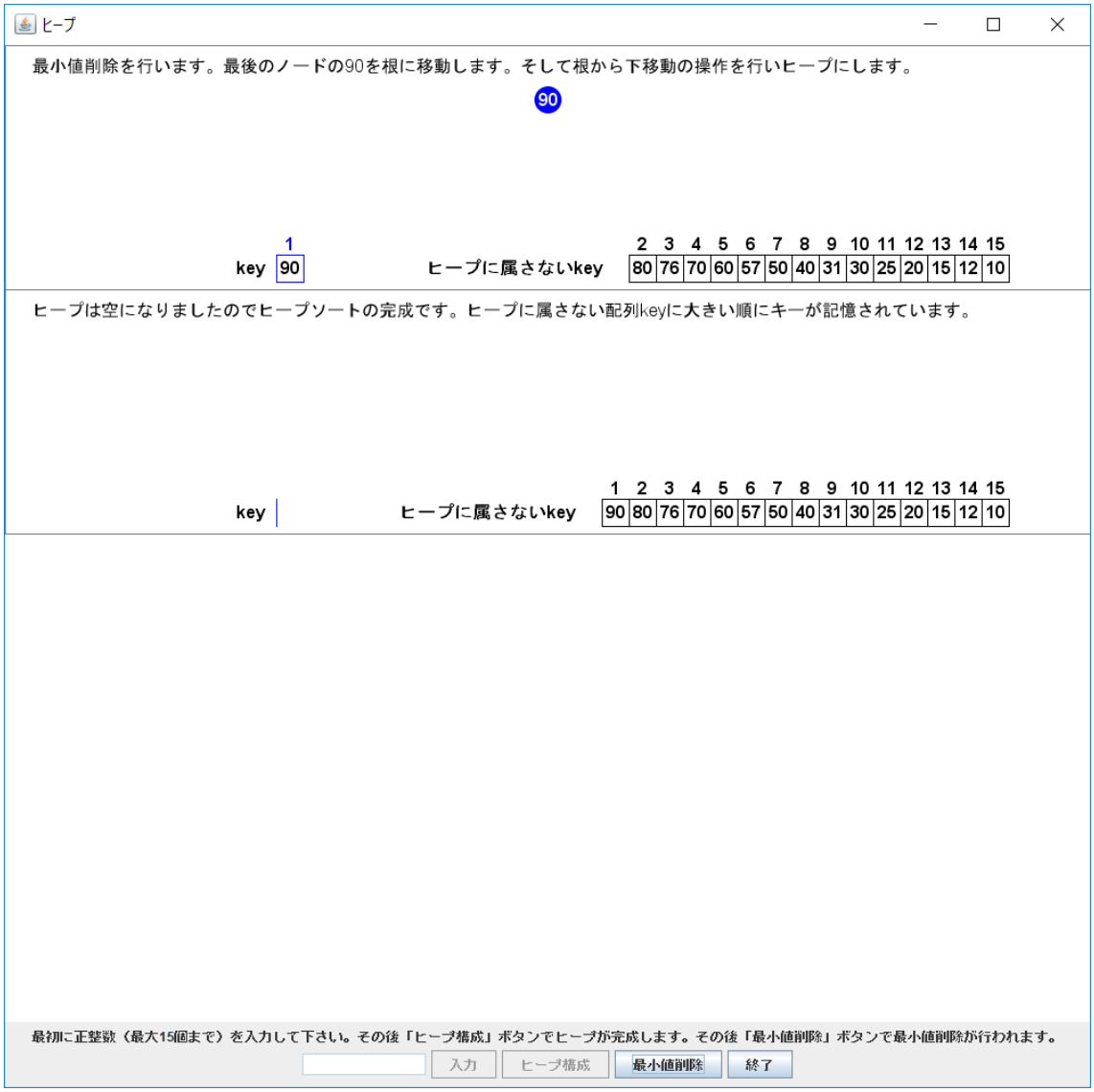
（12 が削除された）

これをこの後 12 回繰り返すとパネルの内容は以下のようになる。



（80 が削除された）

次に（最後になる）「最小値削除」のボタンを押すとパネルの内容は以下のようになる。

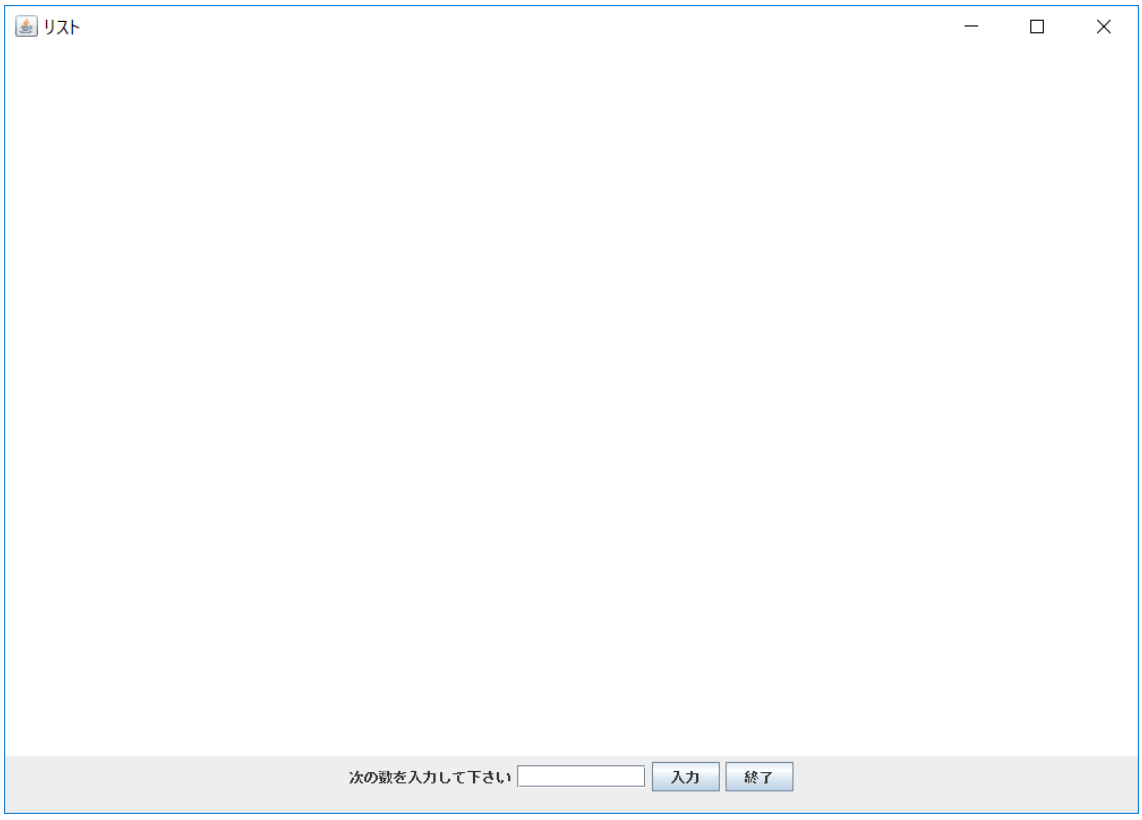


（90 が削除された）

(d) 最後に、「終了」のボタンを押す。

これで、1.のヒープのプログラムの実行は終了して、コマンドプロンプトの画面に、再度
e:¥JavaDemoPrograms>
と表示される。
（途中でうまくいかなかったときには、「終了」ボタンを押すと上記の表示に戻る。）
繰り返したいときは、そこで再度
java HeapDemo 1000 &（とタイプしてその後リターンキーを押す）
として、開かれたパネル上で様々な入力のもとで上記のことを繰り返す。

2. のリスト、3.の二色木、4.の凸包、5.の水平垂直線分の交差列挙の java プログラムの実行もほぼ同様である。以下それらを記す。
まず、リストについて述べる。
e:¥JavaDemoPrograms>
で
java ListDemo 1000 &（とタイプしてその後リターンキーを押す）
とすると、以下のリストのパネルが開かれる。



そこで、各アルファベットが以下のように 2 桁の数字に対応するものとする。

a	b	c	d	e	f	g	h	i	j	k	l	m	n
11	12	13	14	15	16	17	18	19	20	21	22	23	24

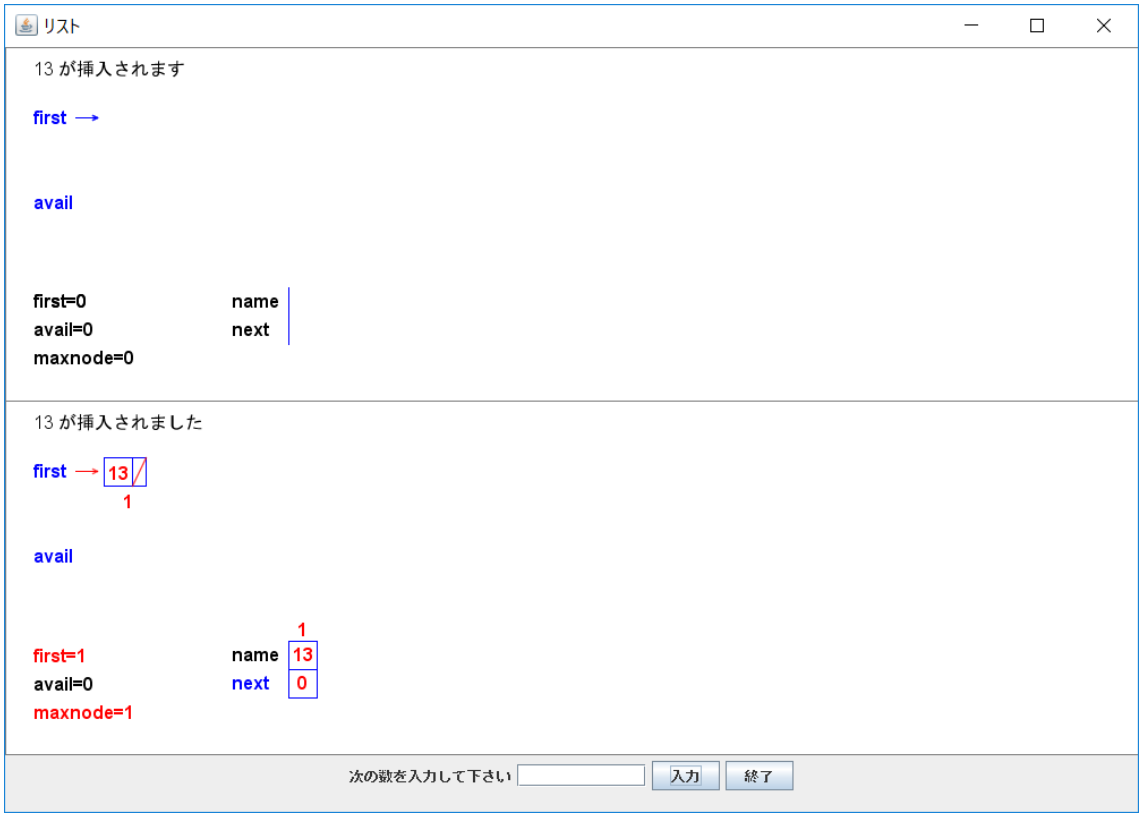
o	p	q	r	s	t	u	v	w	x	y	z
25	26	27	28	29	30	31	32	33	34	35	36

そして、5.6.1 項のように computation を取り上げる。すなわち、
insert(c), insert(o), insert(m), insert(p), insert(u), insert(t),
insert(a), delete(t), insert(i), delete(o), insert(n)
を行う．上の変換を用いると

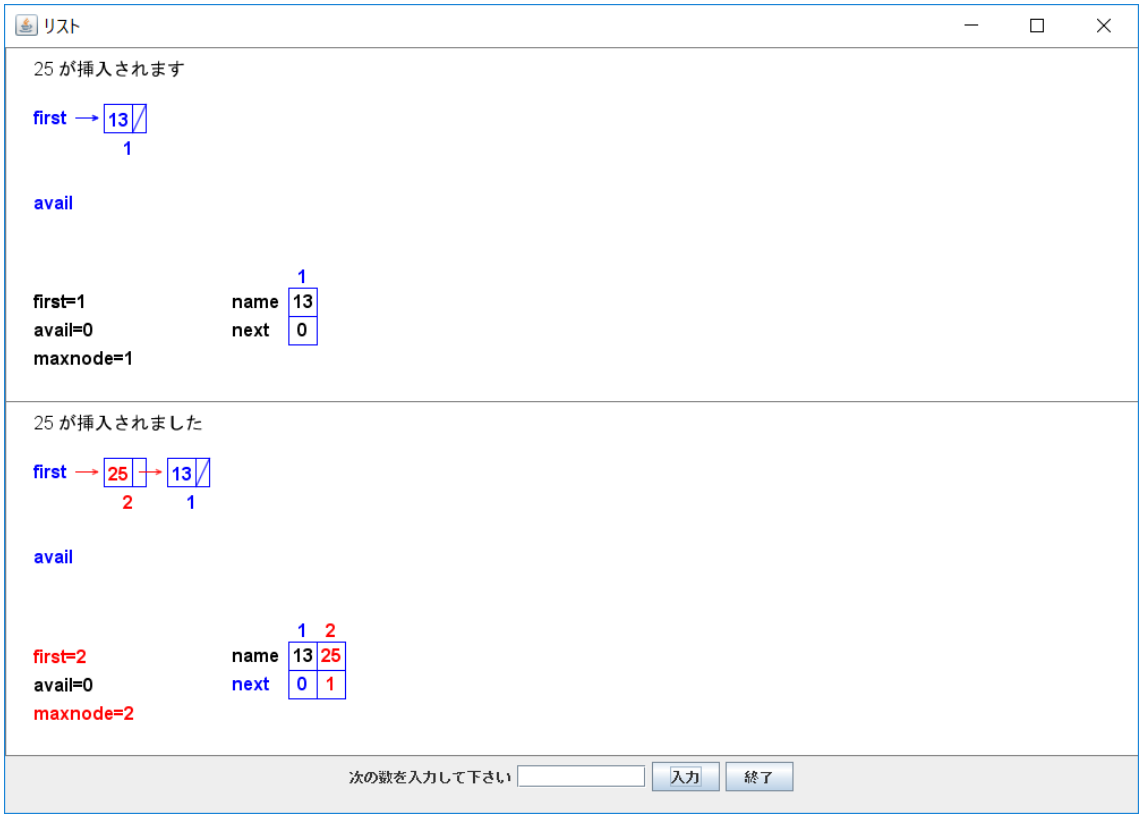
insert(13), insert(25), insert(23), insert(26), insert(31), insert(30),
insert(11), delete(30), insert(19), delete(25), insert(24)
を行うことになる。そこで、半角で

13,25,23,26,31,30,11,30,19,25,24

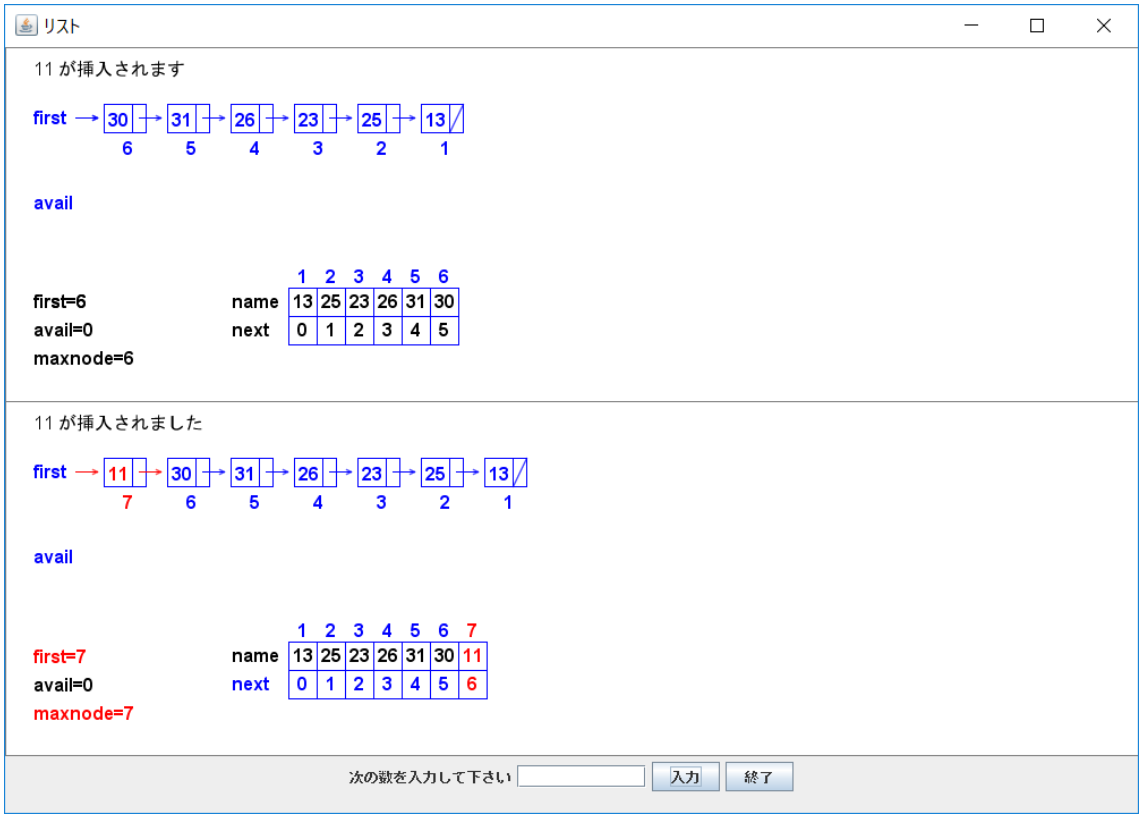
を入力していく。すなわち、パネルは、13 を白い箱の中にタイプして「入力」を押すと



となる。さらに、25 を入力して「入力」を押すと

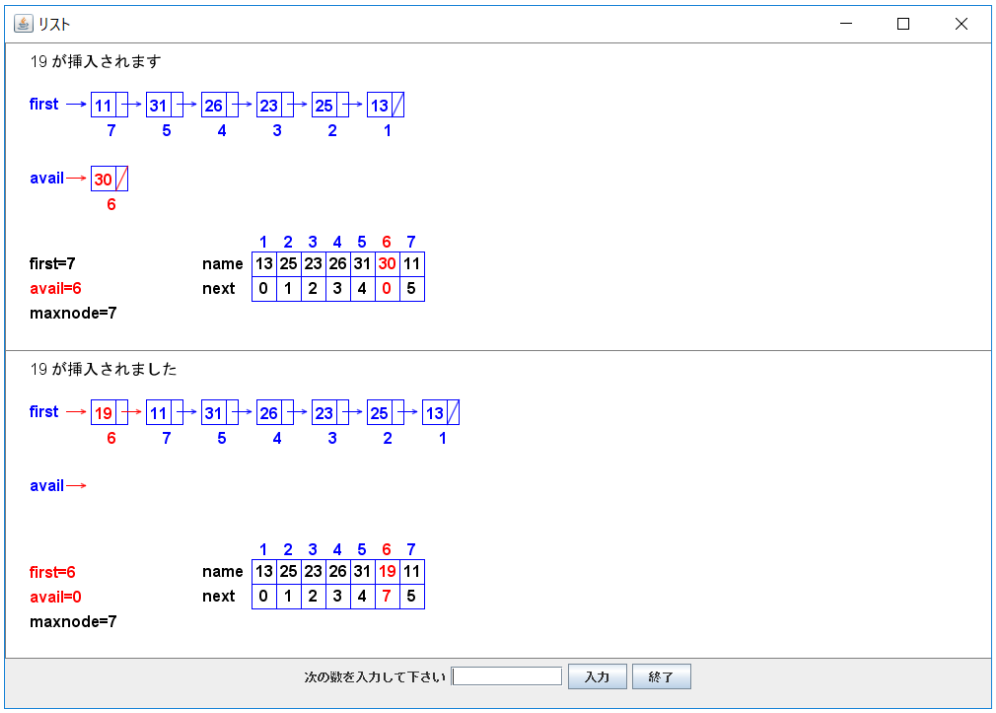


となる。以下同様に続けて、23,26,31,30,11 と入力すると、



となる。そして、30 を入力すると、

となる。さらに、19 を入力すると



となる。さらに、25 を入力すると

さらに、24 を入力すると

となる。これですべての入力が終わったので、「終了」のボタンを押す。すると、コマンドプロンプトの画面に、再度

e:¥JavaDemoPrograms>

と表示される。繰り返したいときは、そこで再度

java ListDemo 1000 &（とタイプしてその後リターンキーを押す）

として、開かれたパネル上で様々な入力のもとで上記のことを繰り返す。

次に、二色木について述べる。

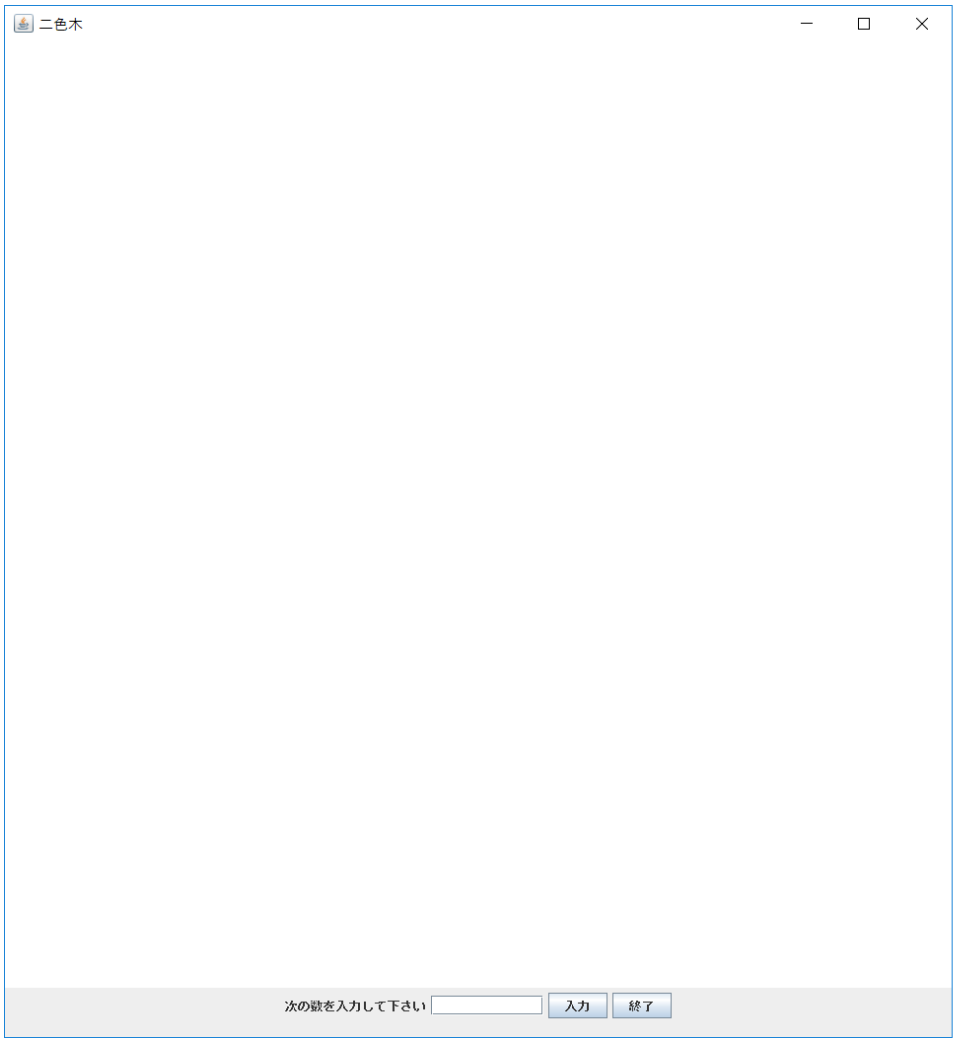
e:¥JavaDemoPrograms>

で

java RBTDemo　960 &（とタイプしてその後リターンキーを押す）

とすると、以下のリストのパネルが開かれる。

（java RBTDemo　1000 &とすると画面にパネルが収まらない可能性がある。）



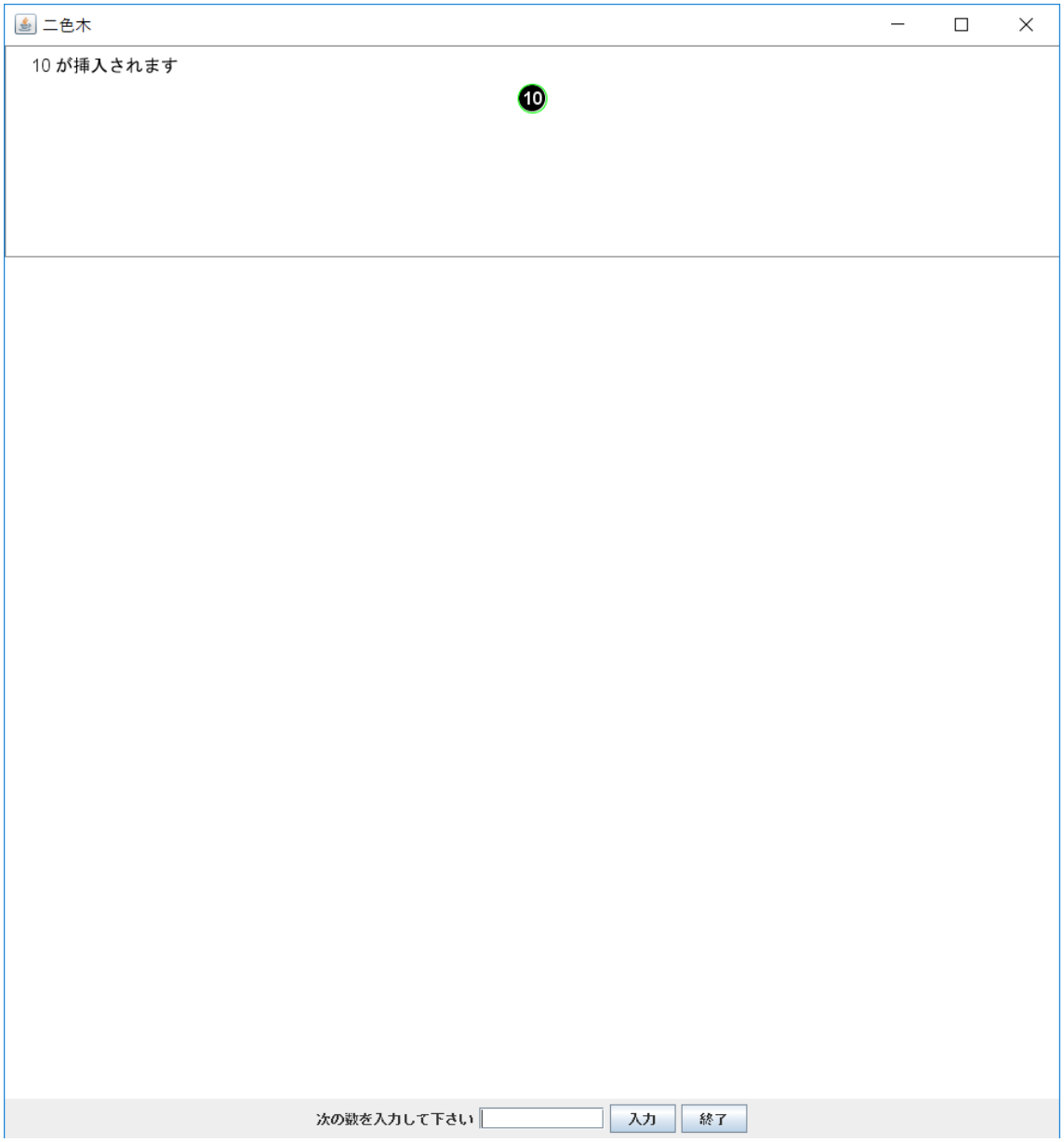
そこで、7.2.3 項で取り上げた系列

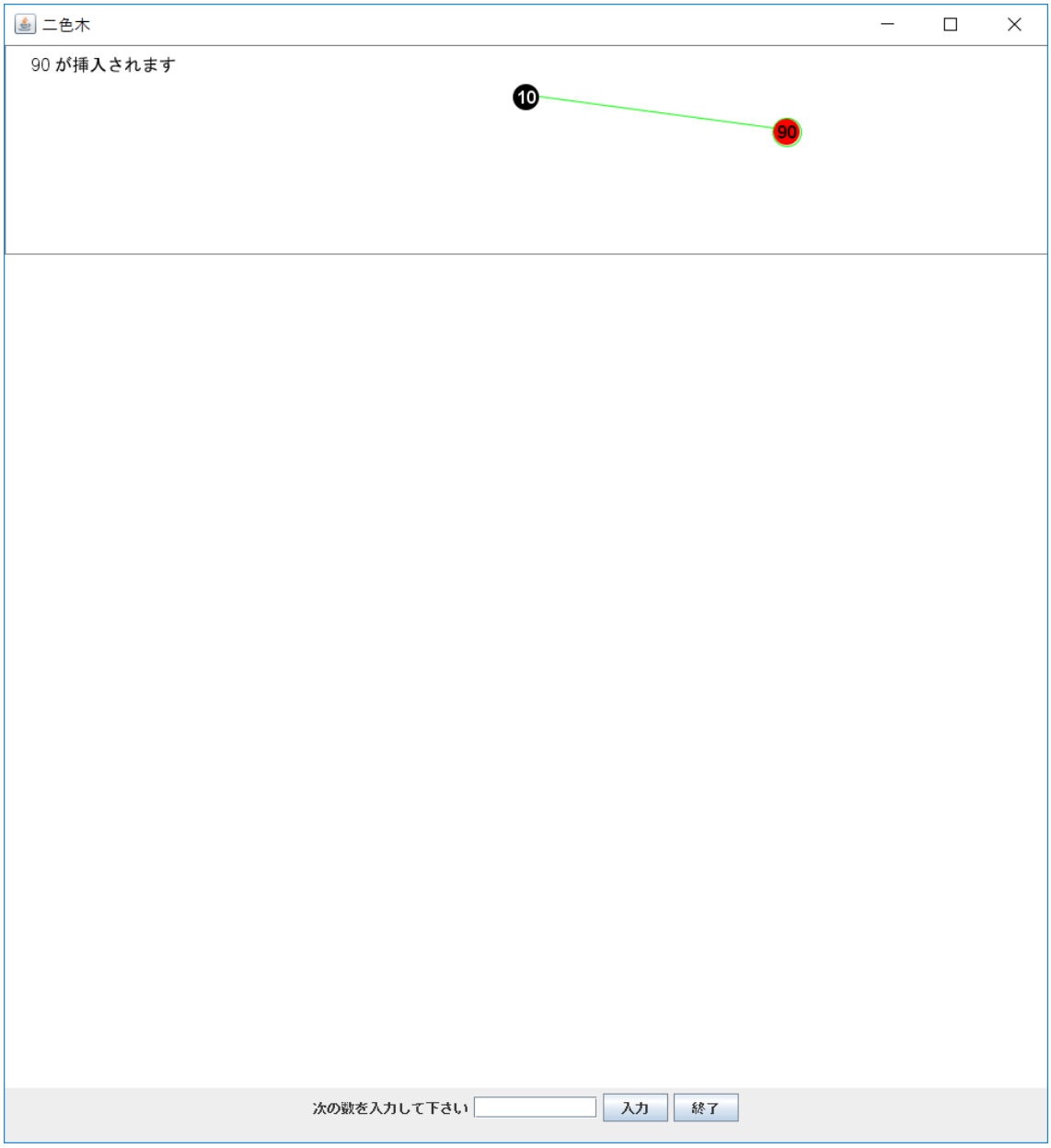
1,9,2,8,3,7,4,6,5,4,3,2,8,9

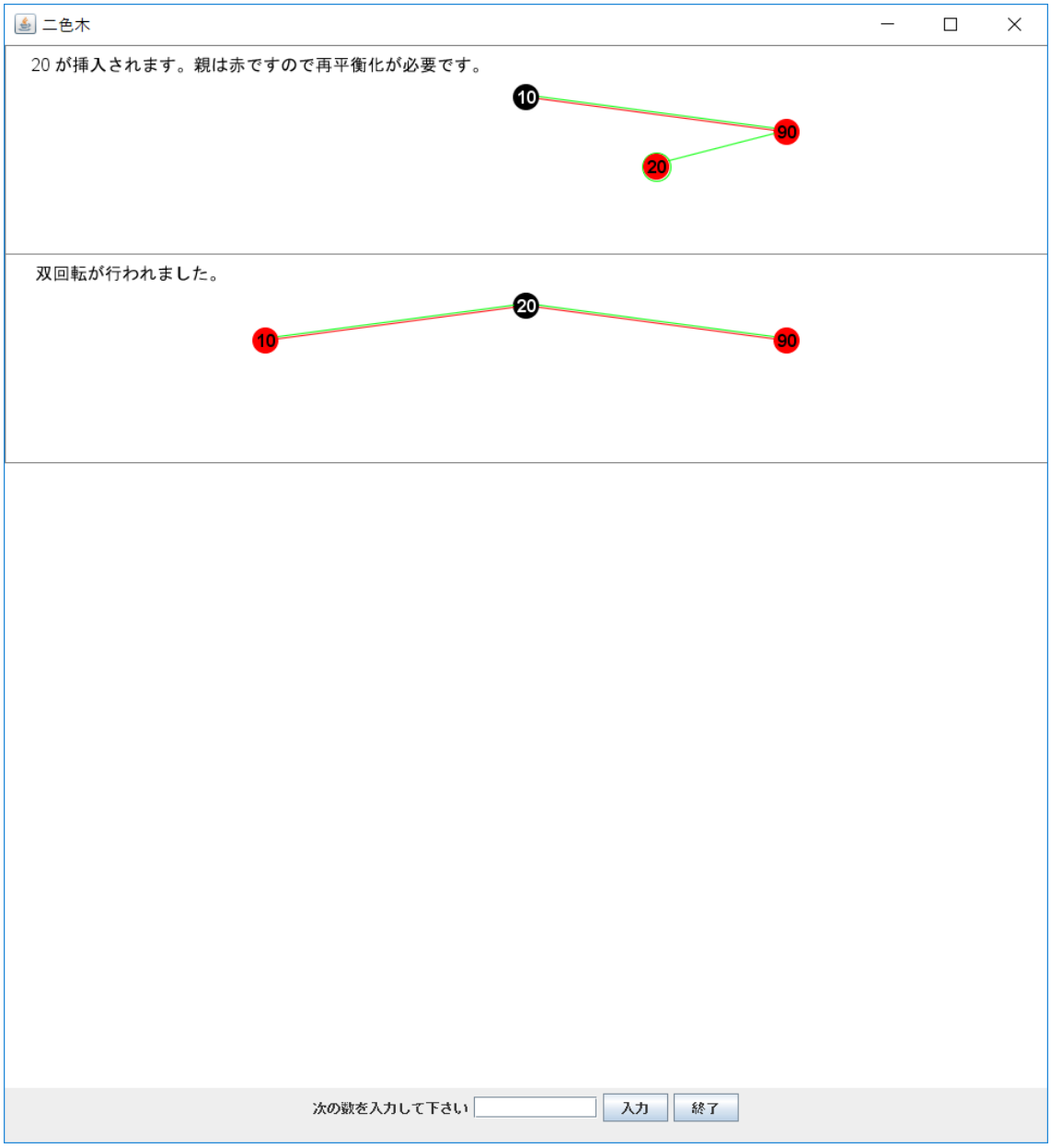
を入力してみる。なお、表示は 2 桁の数字が見やすいように表示するので、実際には

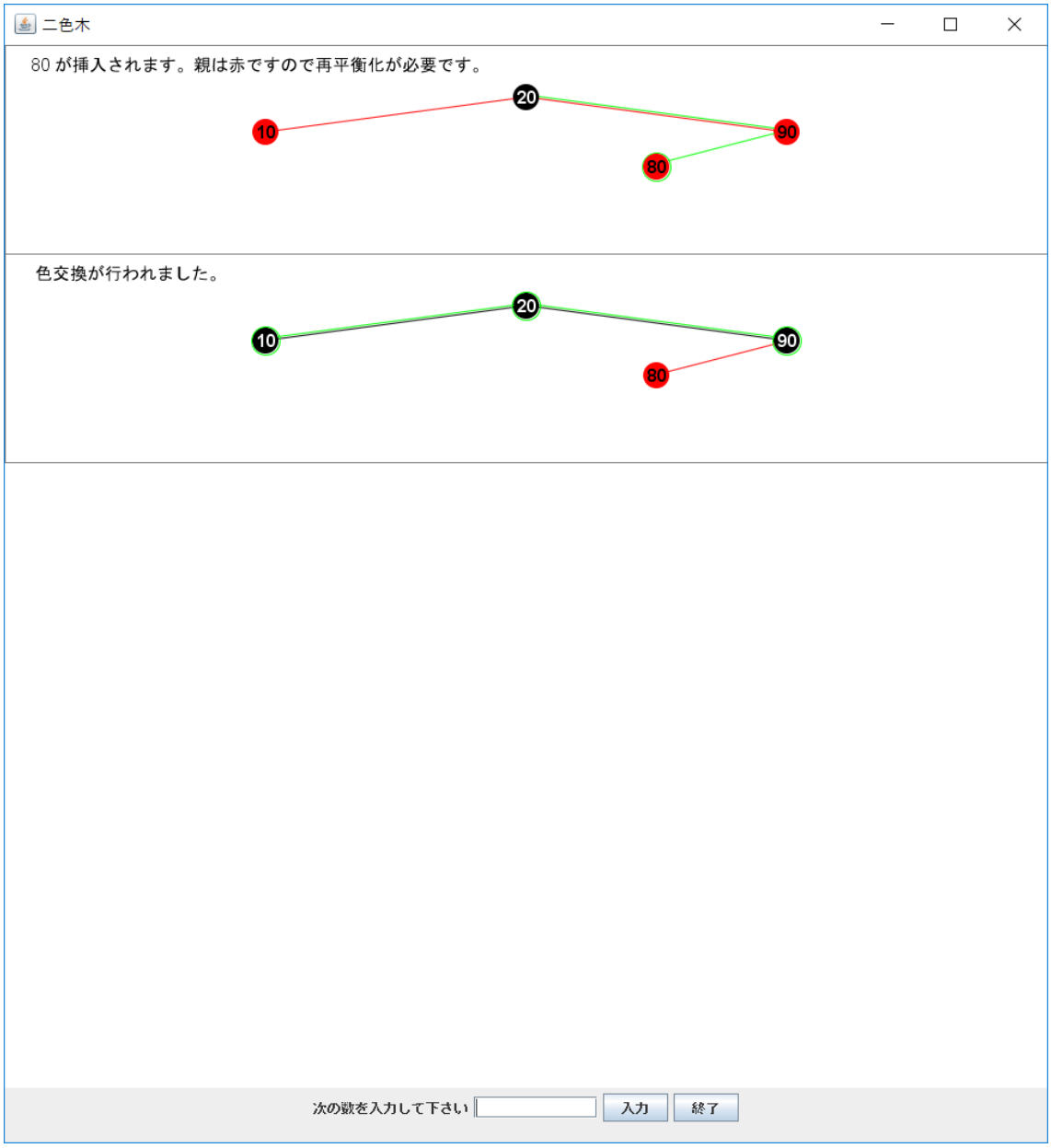
10,90,20,80,30,70,40,60,50,40,30,20,80,90

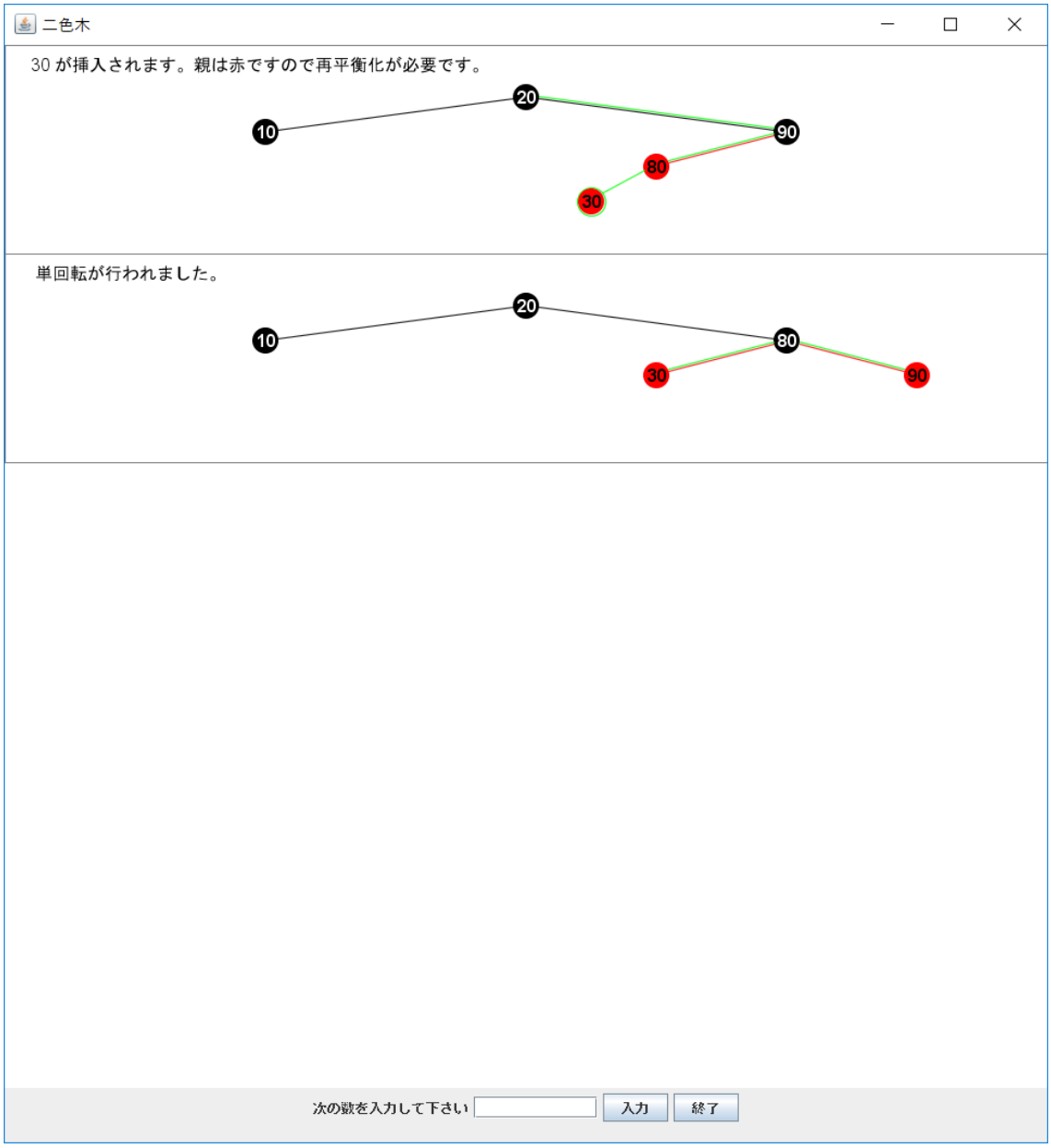
と入力してみる。最初二色木は空であるとして、二色木にない数字が入力されたときには二色木に insert（挿入）され、二色木にあるときには delete（削除）される。以下は得られるパネルの内容の系列である。

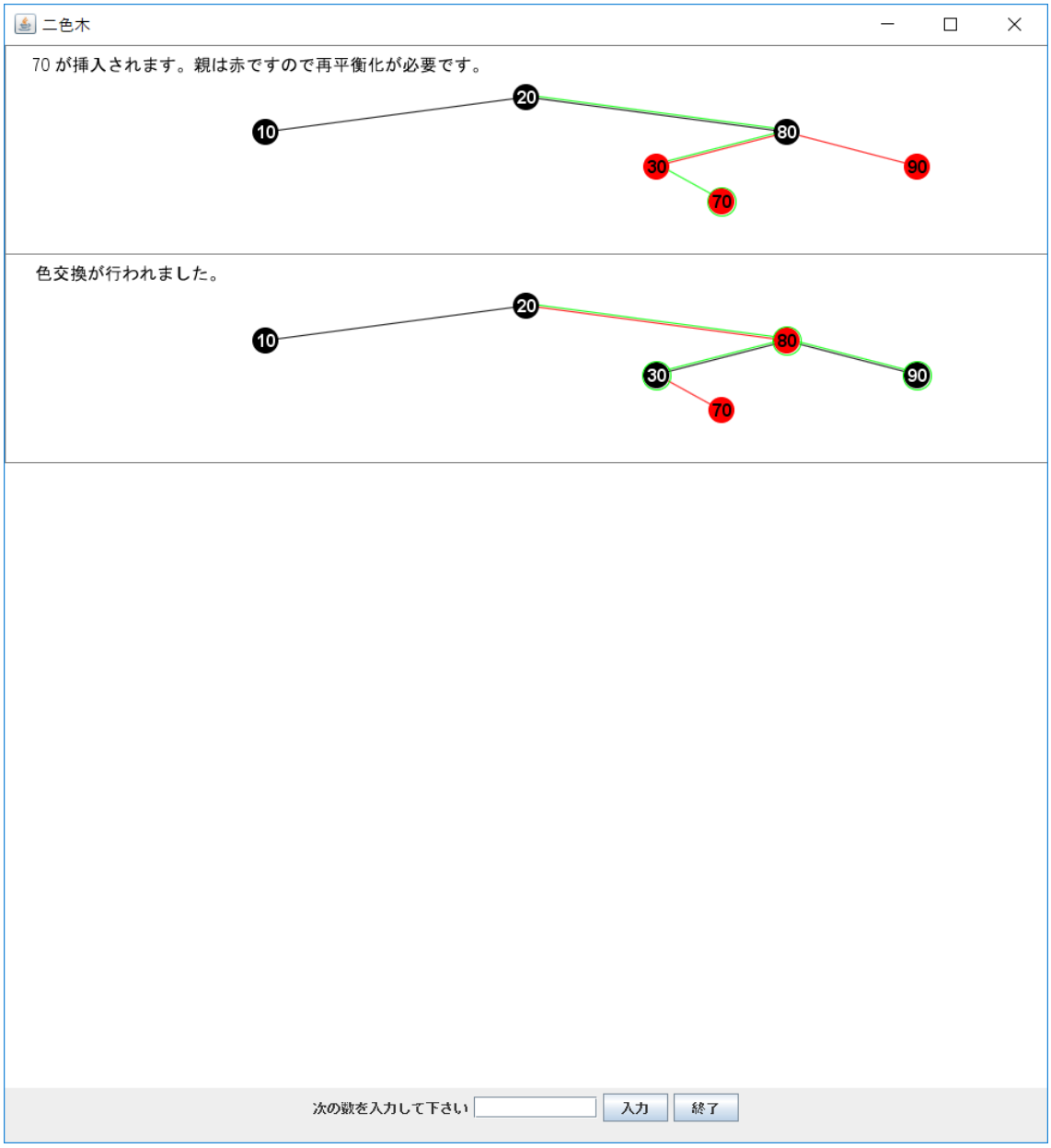

二色木
10 が挿入されます
10
次の数を入力して下さい
入力
終了

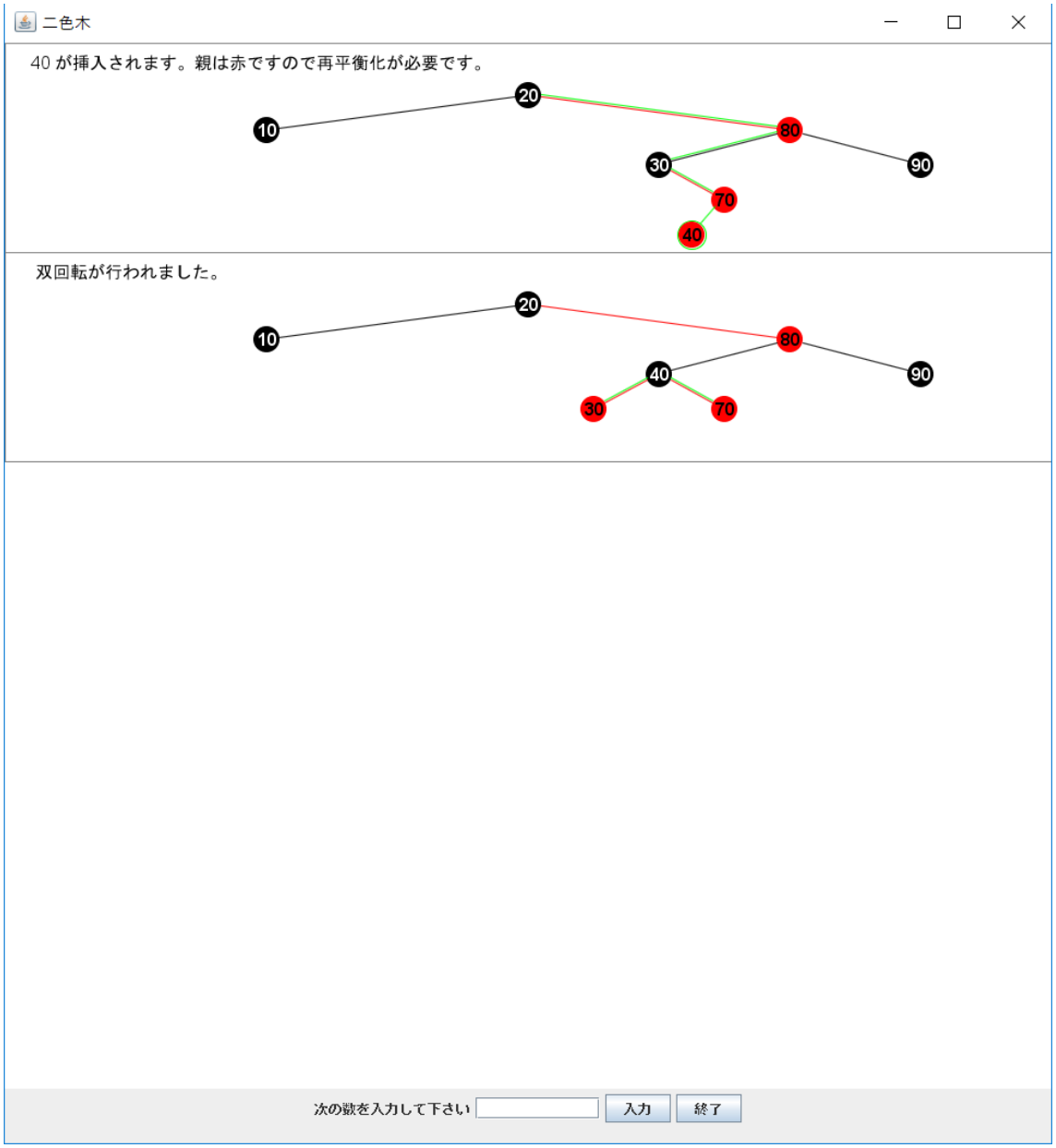

二色木
90 が挿入されます
10
90
次の数を入力して下さい
入力
終了

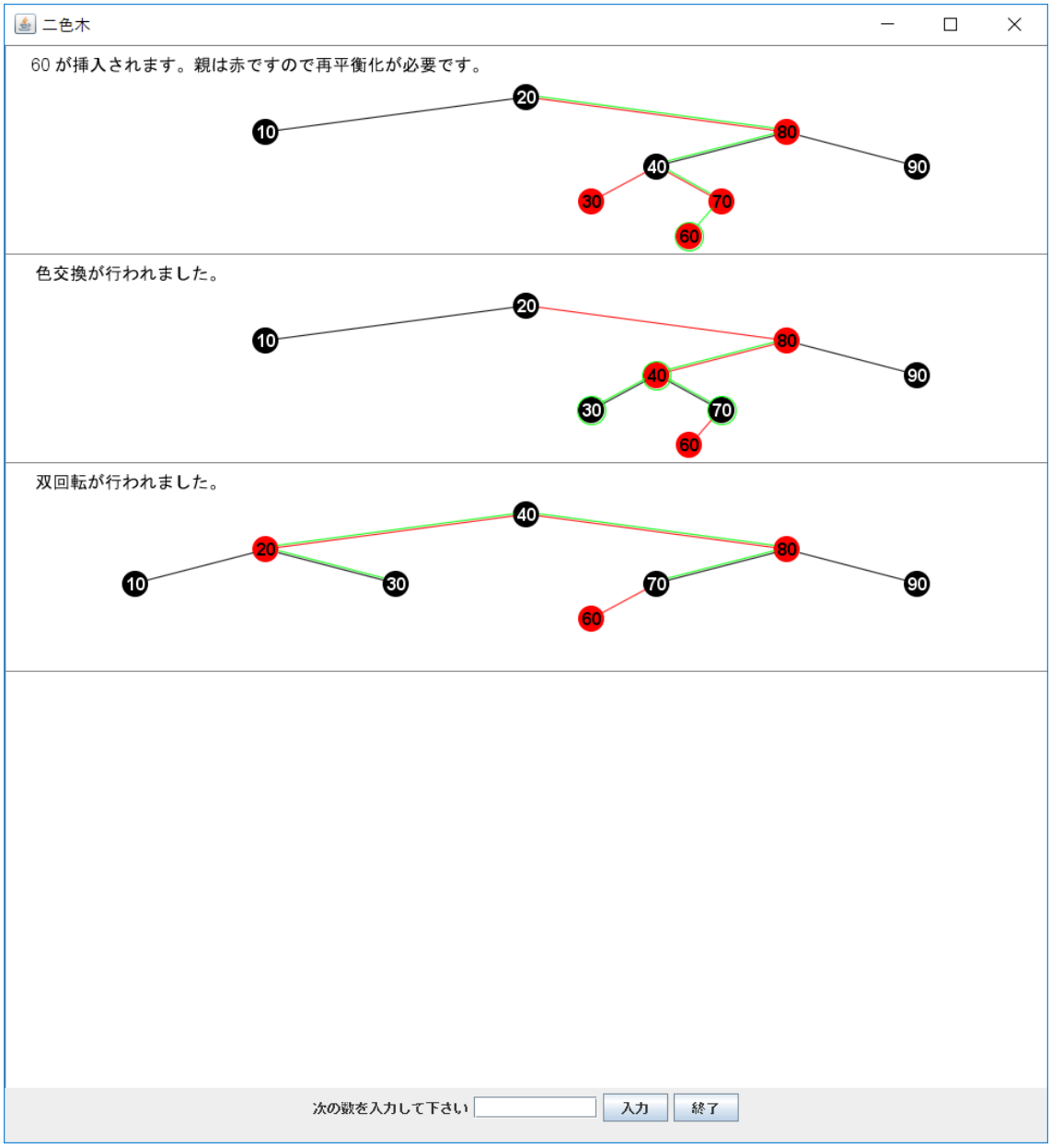

二色木
20 が挿入されます。親は赤ですので再平衡化が必要です。
10
90
20
双回転が行われました。
20
10
90
次の数を入力して下さい
入力
終了

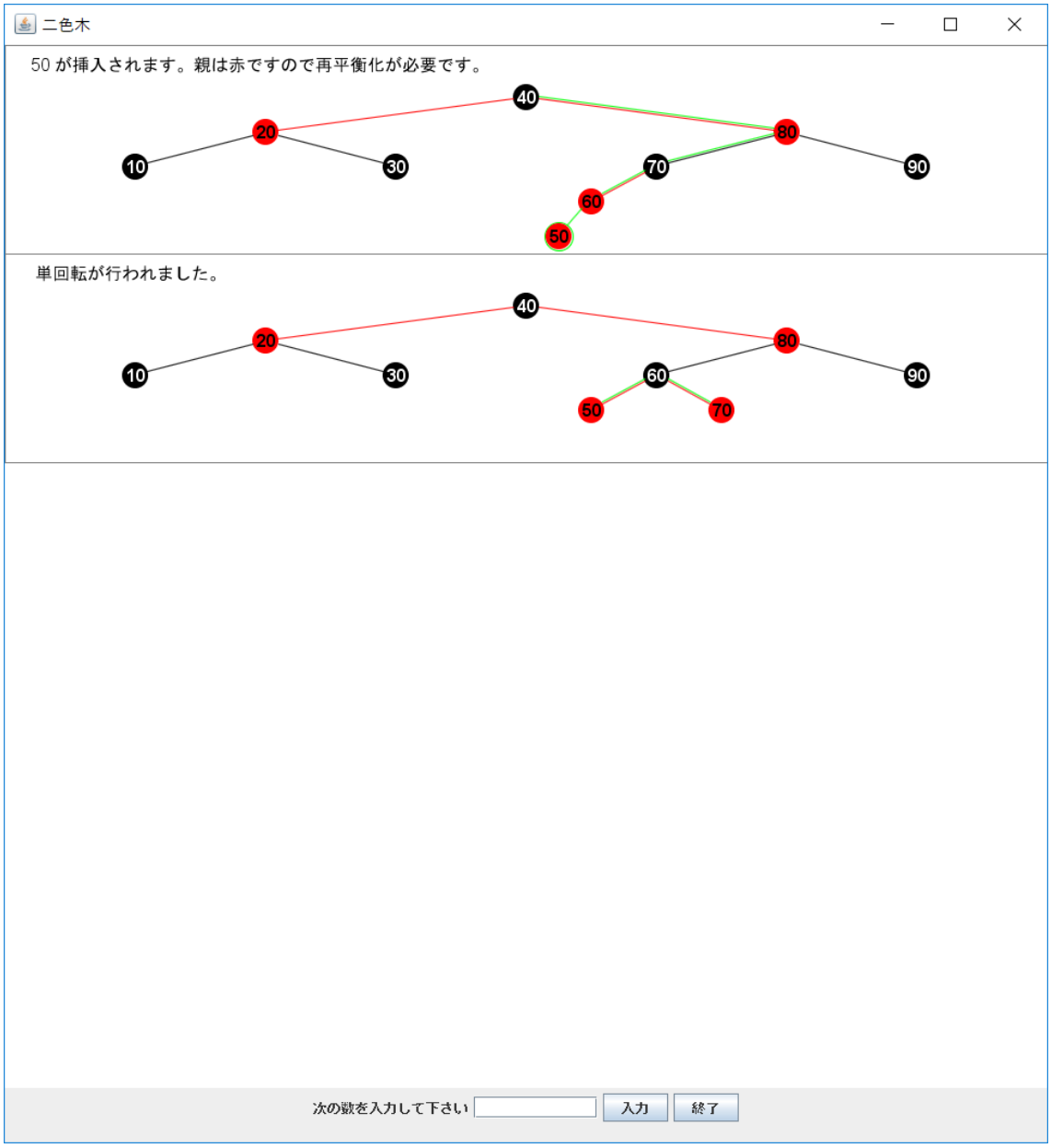

二色木
80 が挿入されます。親は赤ですので再平衡化が必要です。
10
20
80
90
色交換が行われました。
10
20
80
90
次の数を入力して下さい
入力
終了

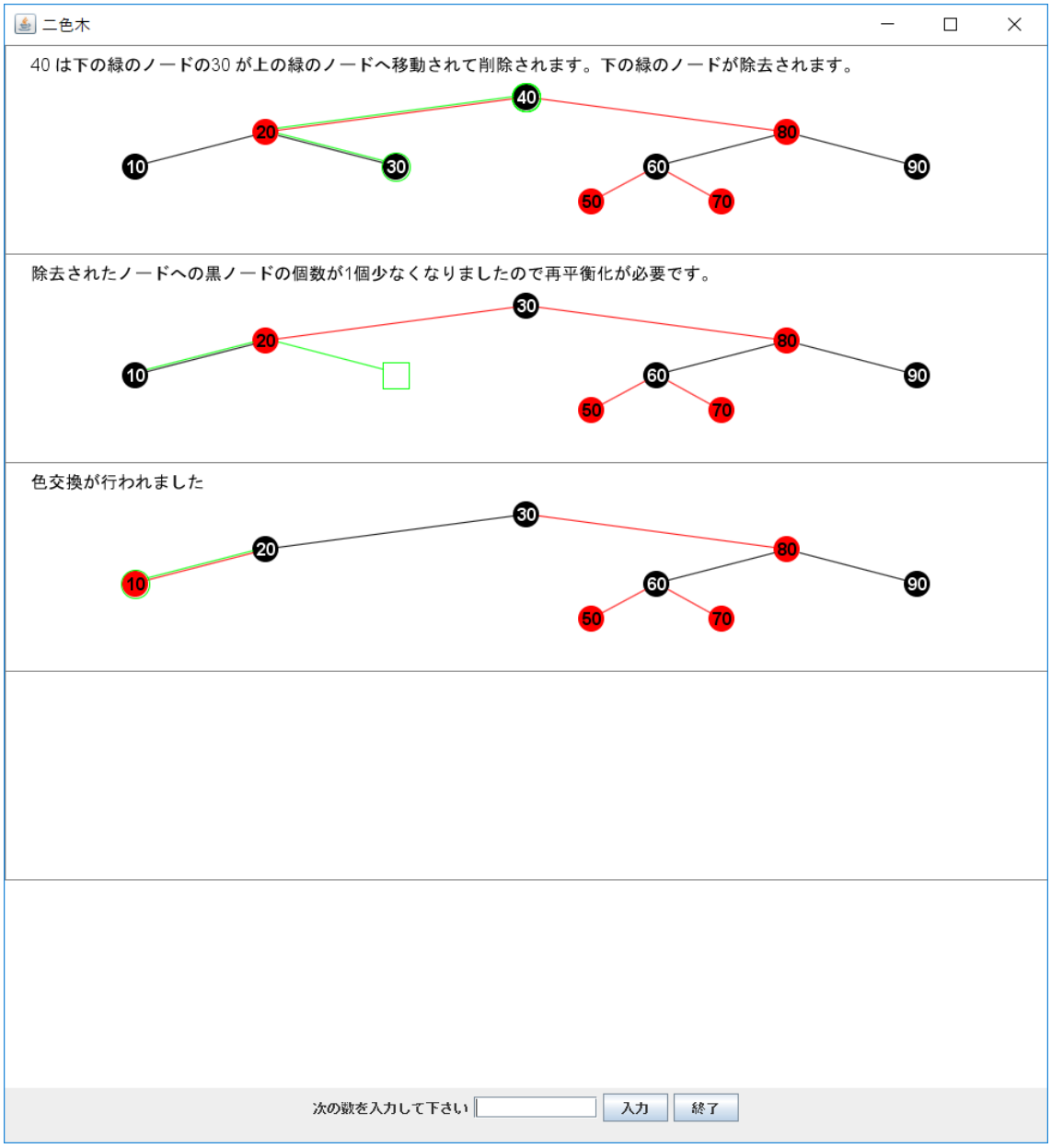

二色木
30 が挿入されます。親は赤ですので再平衡化が必要です。
10
20
90
80
30
単回転が行われました。
10
20
80
30
90
次の数を入力して下さい
入力
終了

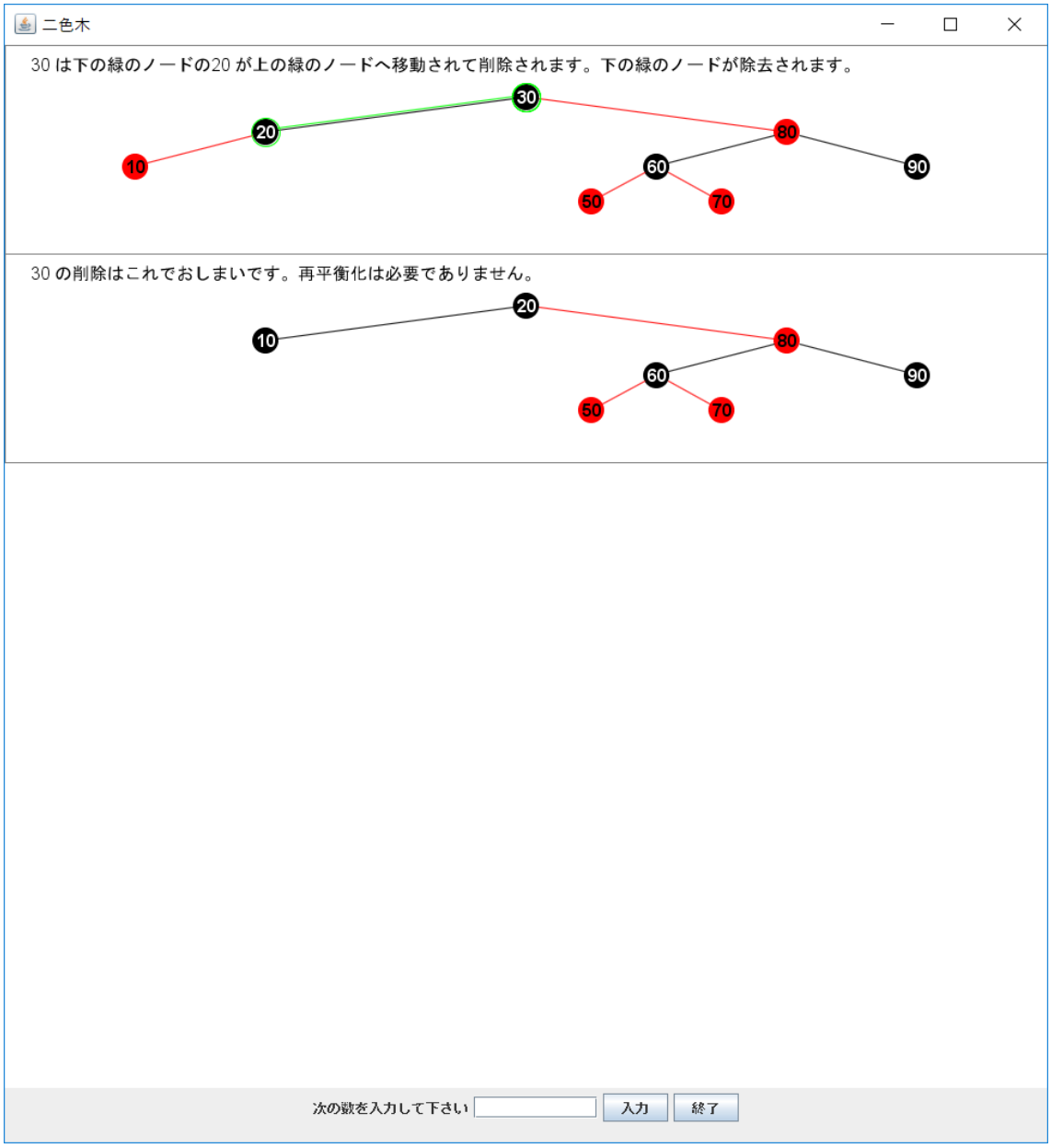

二色木
70 が挿入されます。親は赤ですので再平衡化が必要です。
10
20
30
70
80
90
色交換が行われました。
10
20
30
70
80
90
次の数を入力して下さい
入力
終了

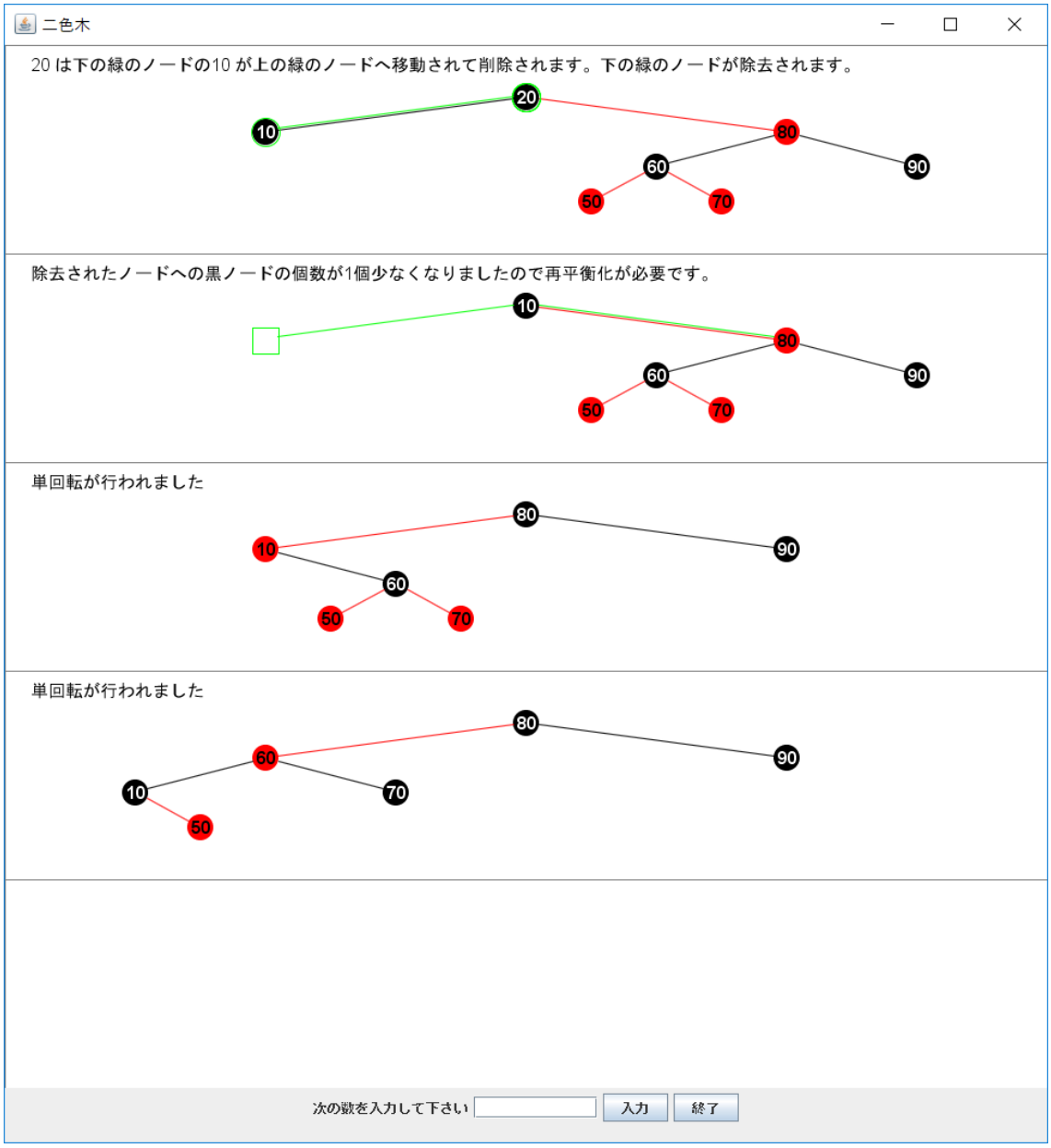

二色木
40 が挿入されます。親は赤ですので再平衡化が必要です。
20
10
80
30
90
70
40
双回転が行われました。
20
10
80
40
90
30
70
次の数を入力して下さい
入力
終了

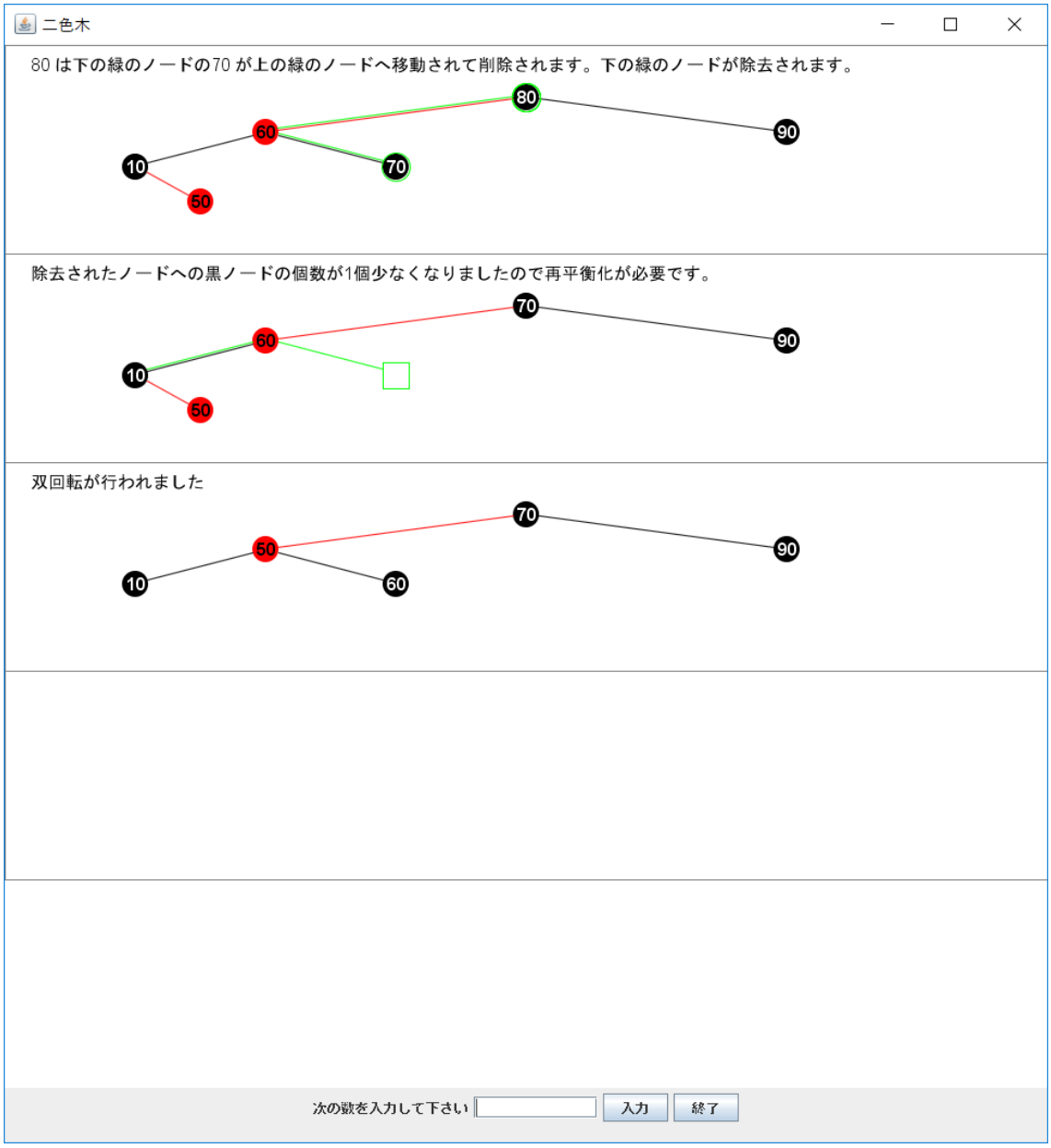

二色木
60 が挿入されます。親は赤ですので再平衡化が必要です。
20
10
80
40
90
30
70
60
色交換が行われました。
20
10
80
40
90
30
70
60
双回転が行われました。
40
20
80
10
30
70
90
60
次の数を入力して下さい
入力
終了

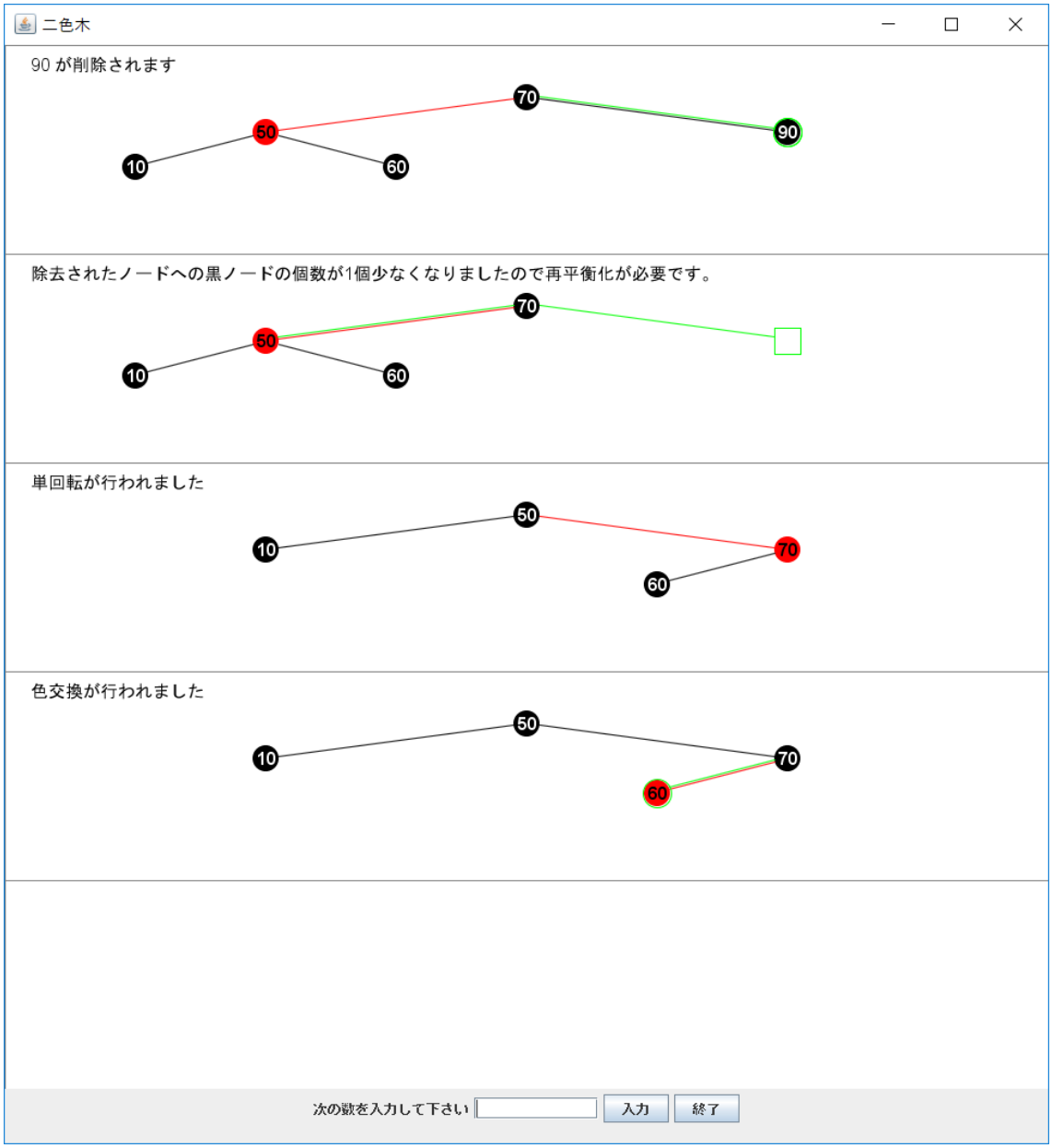

二色木
50 が挿入されます。親は赤ですので再平衡化が必要です。
40
20
80
10
30
70
90
60
50
単回転が行われました。
40
20
80
10
30
60
90
50
70
次の数を入力して下さい
入力
終了


二色木
40 は下の緑のノードの30 が上の緑のノードへ移動されて削除されます。下の緑のノードが除去されます。
40
20
80
10
30
60
90
50
70
除去されたノードへの黒ノードの個数が1個少なくなりましたので再平衡化が必要です。
30
20
80
10
60
90
50
70
色交換が行われました
30
20
80
10
60
90
50
70
次の数を入力して下さい
入力
終了


二色木
30 は下の緑のノードの20 が上の緑のノードへ移動されて削除されます。下の緑のノードが除去されます。
30
20
80
10
60
90
50
70
30 の削除はこれでおしまいです。再平衡化は必要でありません。
20
10
80
60
90
50
70
次の数を入力して下さい
入力
終了

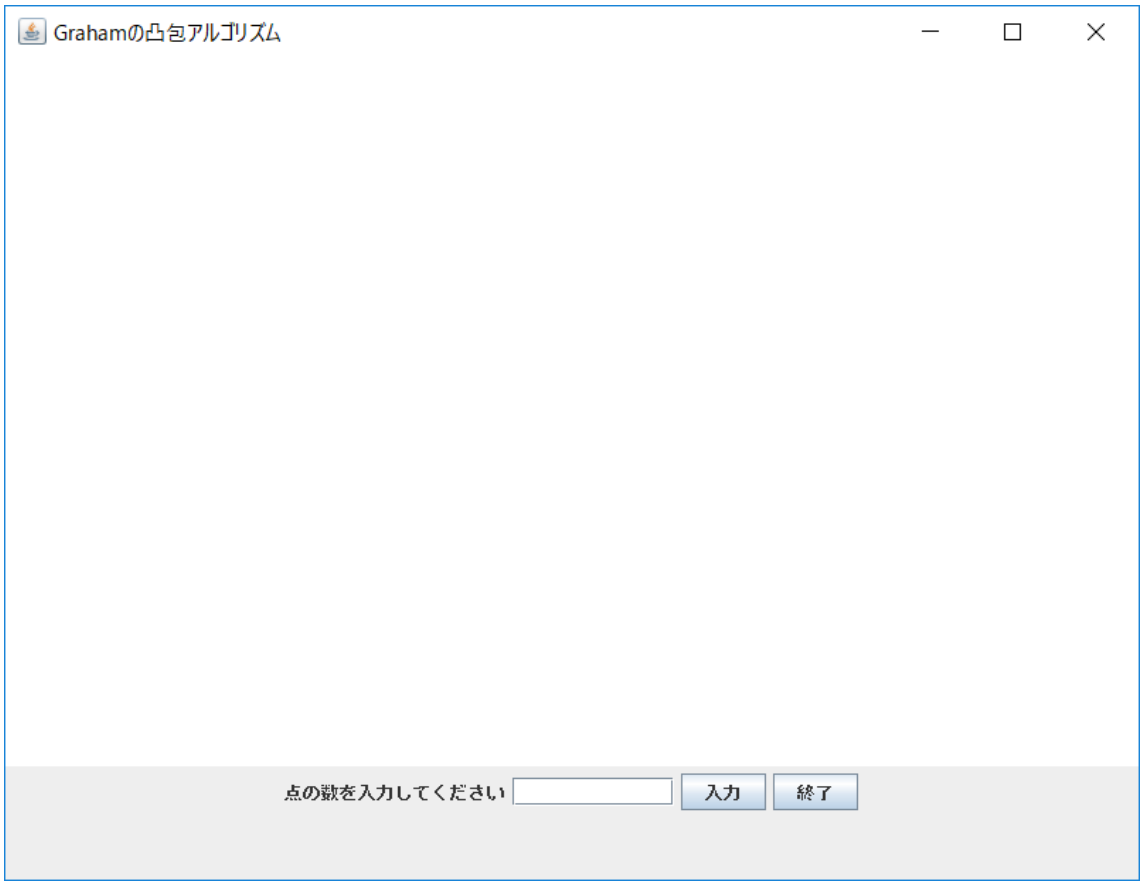

二色木
20 は下の緑のノードの10 が上の緑のノードへ移動されて削除されます。下の緑のノードが除去されます。
20
10
80
60
90
50
70
除去されたノードへの黒ノードの個数が1個少なくなりましたので再平衡化が必要です。
10
80
60
90
50
70
単回転が行われました
80
10
90
60
50
70
単回転が行われました
80
60
90
10
70
50
次の数を入力して下さい
入力
終了


二色木
80 は下の緑のノードの70 が上の緑のノードへ移動されて削除されます。下の緑のノードが除去されます。
80
60
90
10
70
50
除去されたノードへの黒ノードの個数が1個少なくなりましたので再平衡化が必要です。
70
60
90
10
50
双回転が行われました
70
50
90
10
60
次の数を入力して下さい
入力
終了



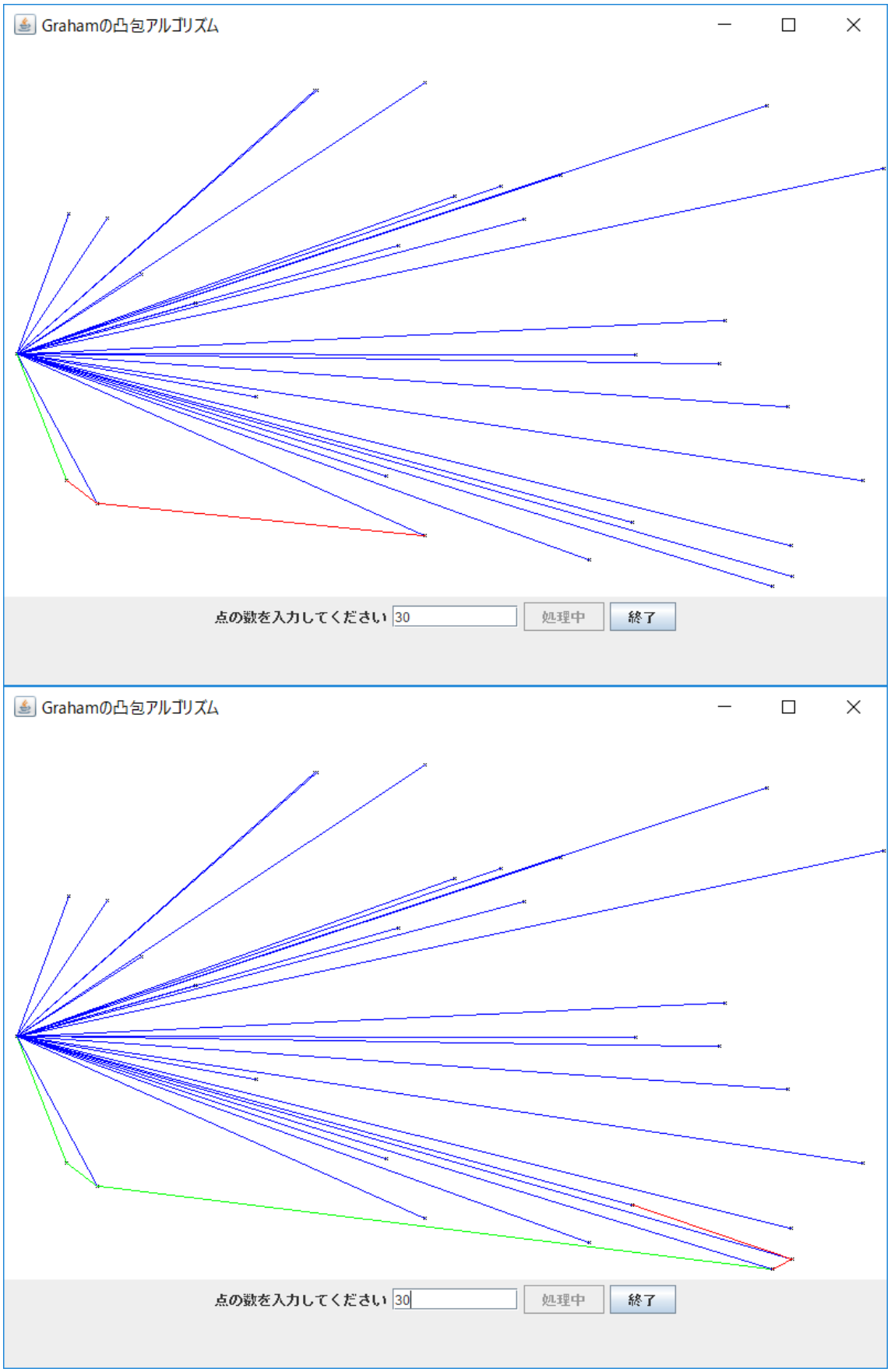
終了するときには「終了」のボタンを押す。すると、コマンドプロンプトの画面に、再度
e:¥JavaDemoPrograms>
と表示される。繰り返したいときは、そこで再度
java RBTDemo 960 &（とタイプしてその後リターンキーを押す）
として、開かれたパネル上で様々な入力のもとで上記のことを繰り返す。

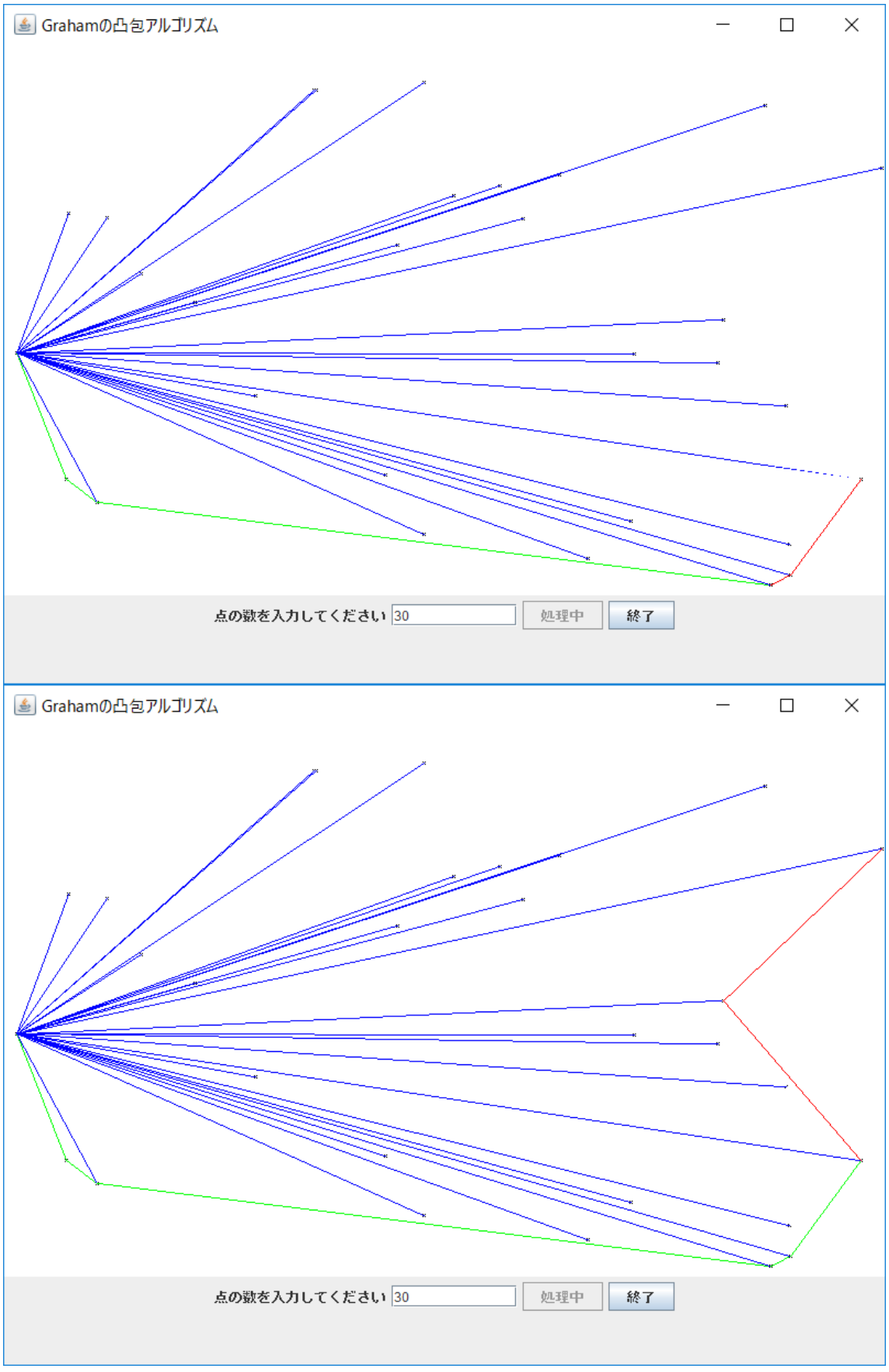
次に、凸包（Graham の凸包を求めるアルゴリズム）について述べる。
e:¥JavaDemoPrograms>
で
java GrahamDemo 800 &（とタイプしてその後リターンキーを押す）
とすると、以下の凸包のパネルが開かれる。

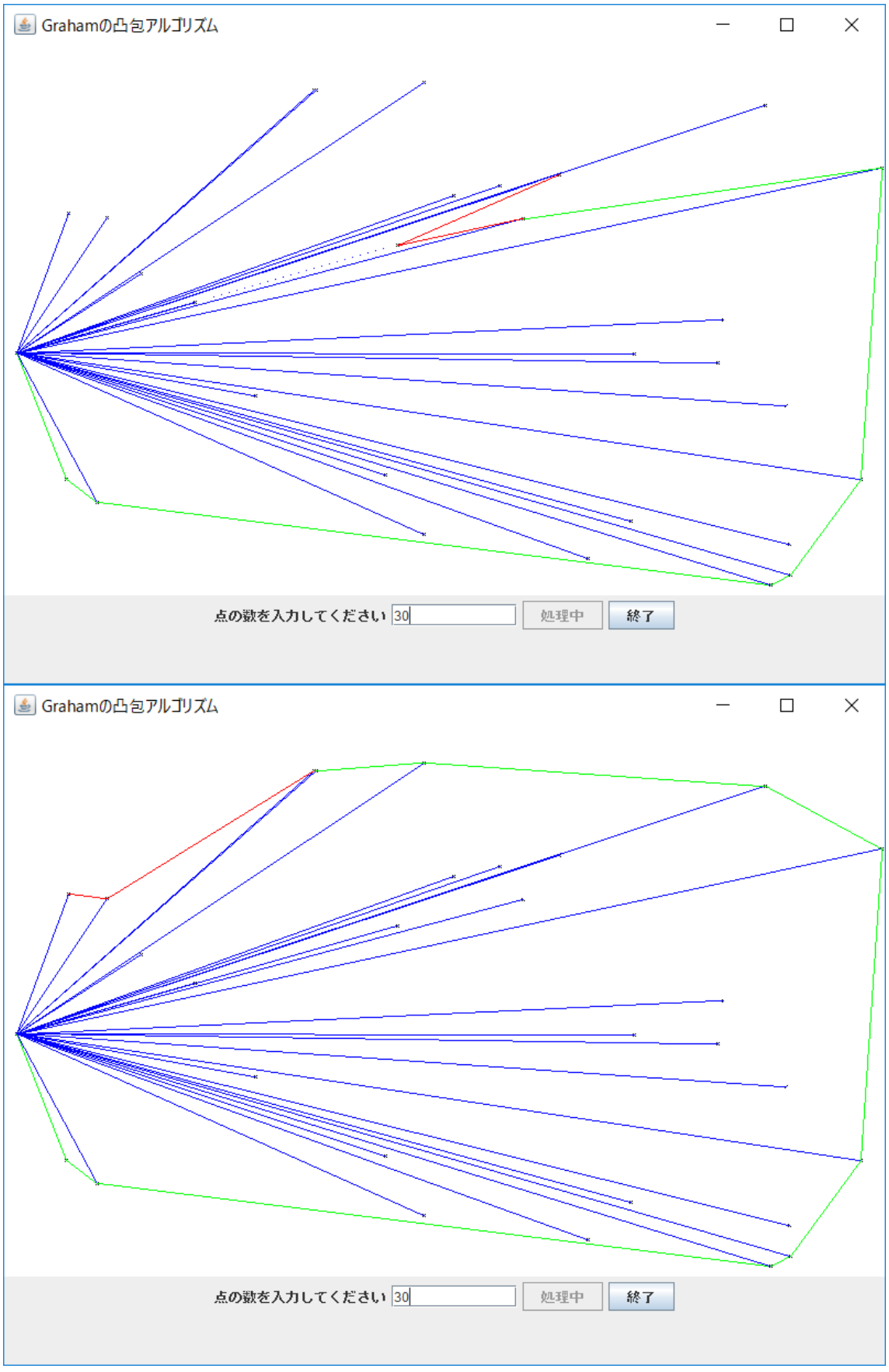


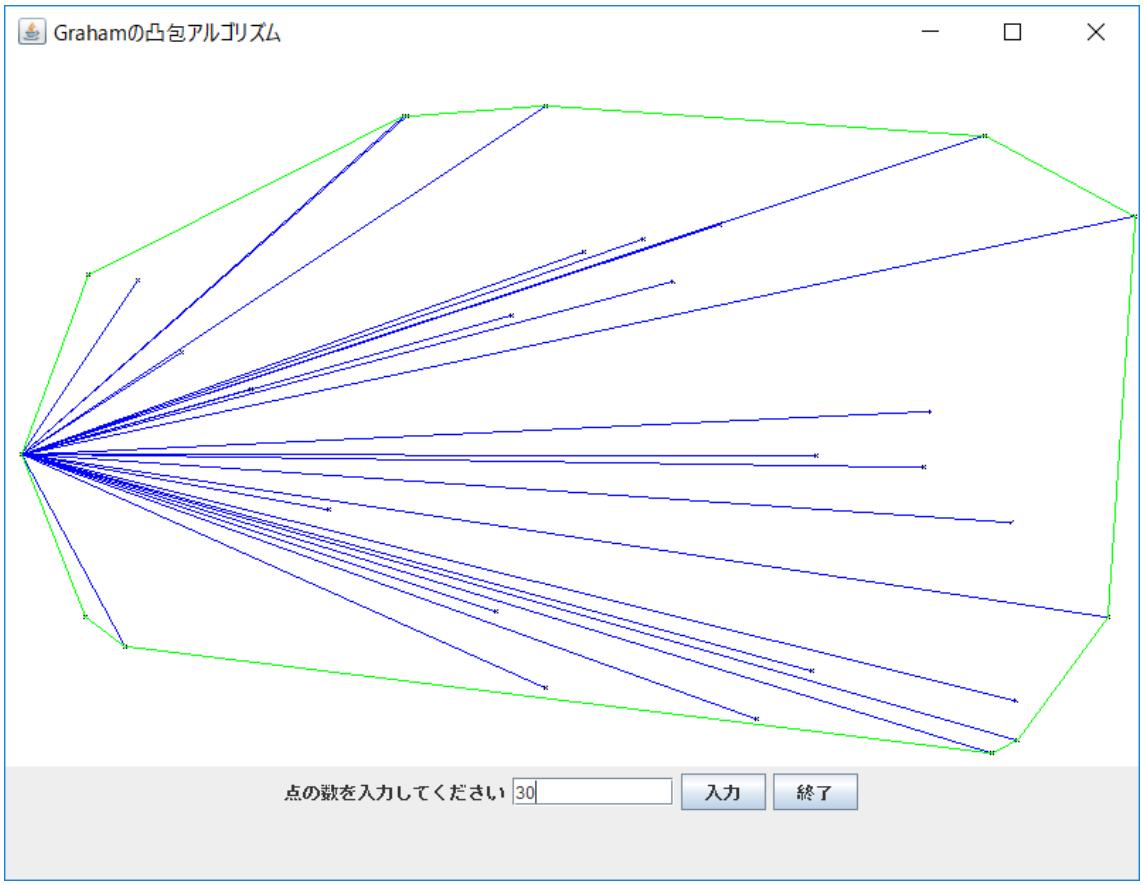
ここで、点数を入力する（点数を記入して「入力」のボタンを押す)。
すると、以下のようなパネルの更新が観察できる（点数は 30 として入力している)。

最初、最も左の点から各点へ偏角の小さい順に青い線を引いている。偏角順に正しくソートされていることがわかる。次に、偏角順に赤い線で表示している内角が 180 度未満かどうかを調べる。内角が 180 度未満のときにはその内角の点まで緑の線を延ばす。一方、内角が 180 度以上のときには、その点を削除して、先端の緑の線を赤に戻して一つ前の内角を調べに戻る。その様子（Graham 走査の様子）を赤い線で表示している。最後に、緑の線が凸包を表している。なお、Graham 走査を目で見て確かめられるように、次のステップに移るまで適切と思われる待ち時間を設定している。


Grahamの凸包アルゴリズム
点の数を入力してください
30
処理中
終了
Grahamの凸包アルゴリズム
点の数を入力してください
30
処理中
終了

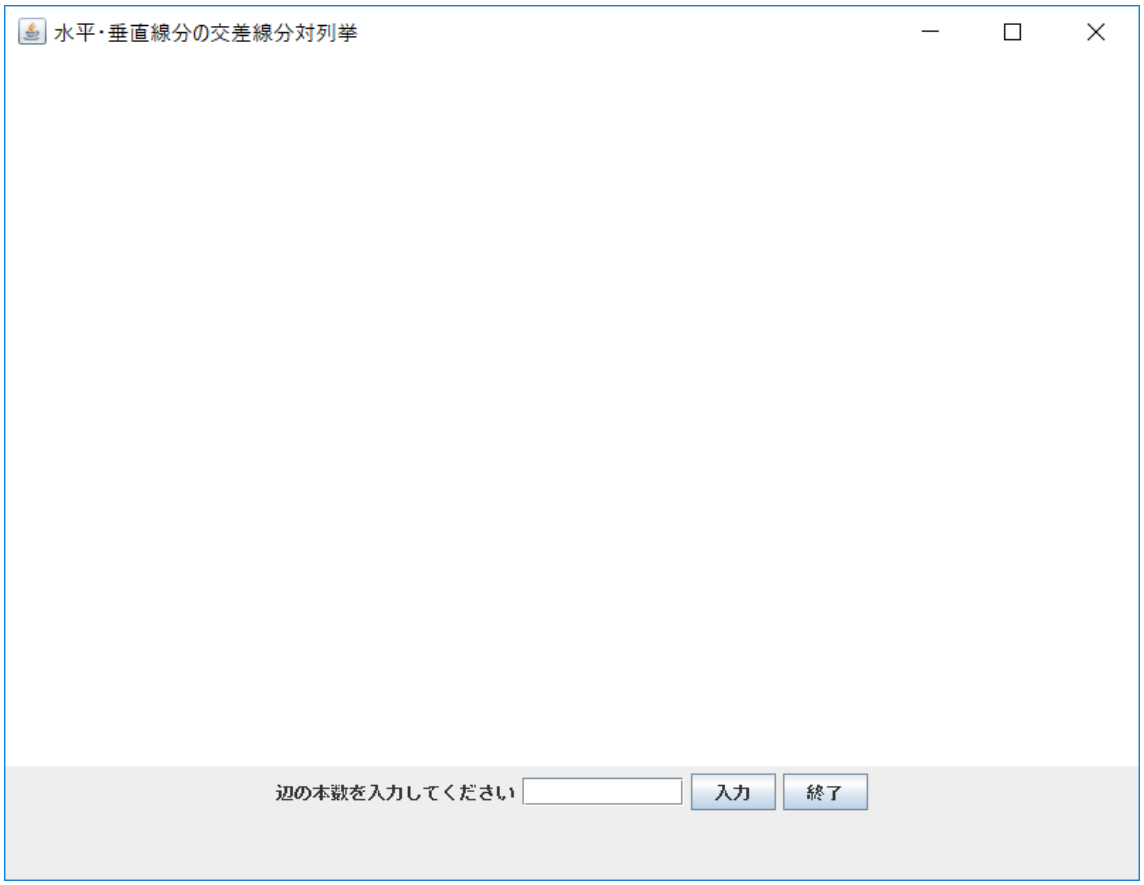

Grahamの凸包アルゴリズム
点の数を入力してください
30
処理中
終了
Grahamの凸包アルゴリズム
点の数を入力してください
30
処理中
終了


Grahamの凸包アルゴリズム
点の数を入力してください
30
処理中
終了
Grahamの凸包アルゴリズム
点の数を入力してください
30
処理中
終了



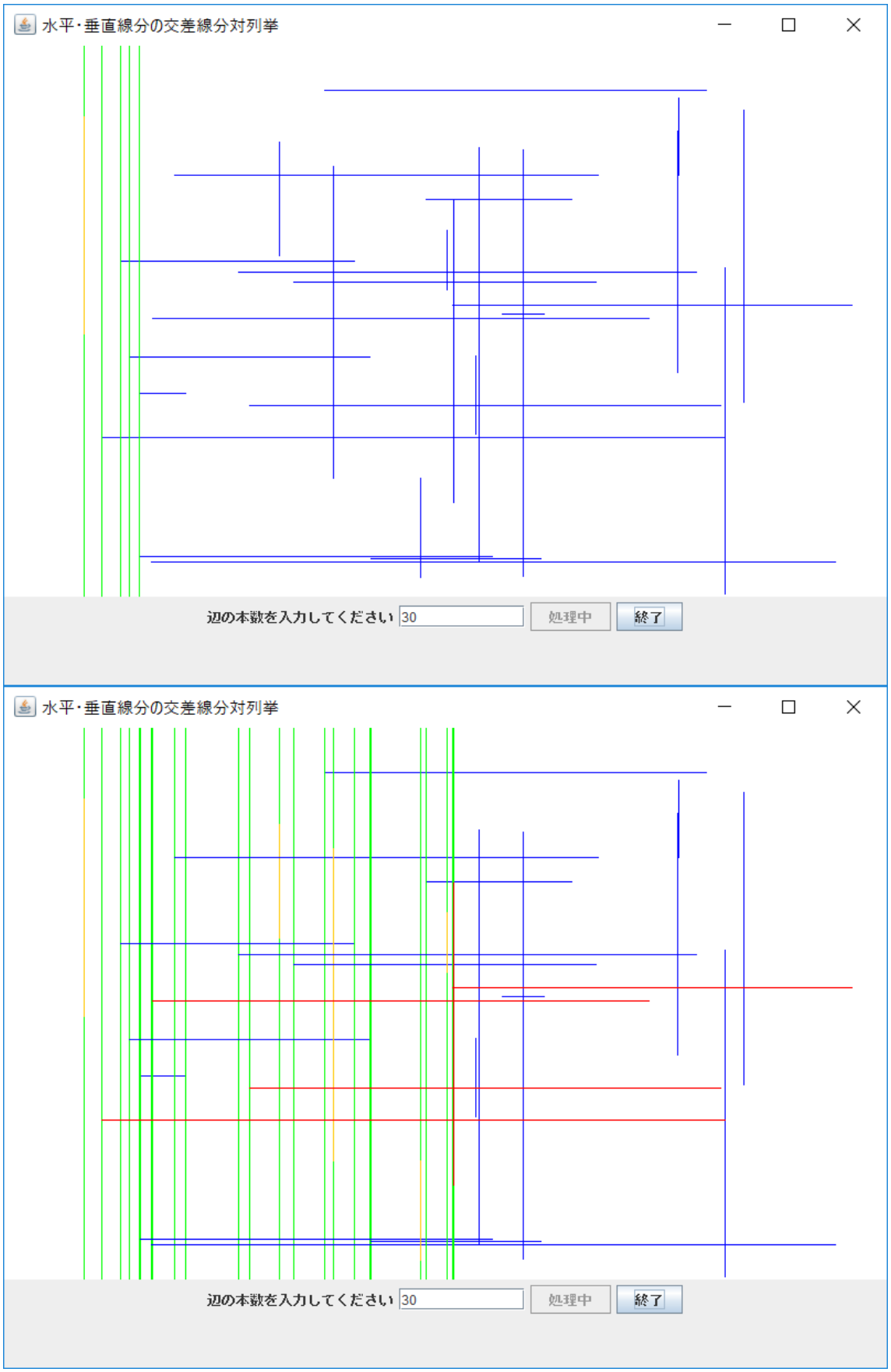
終了するときには「終了」のボタンを押す。すると、コマンドプロンプトの画面に、再度
e:¥JavaDemoPrograms>
と表示される。繰り返したいときは、そこで再度
java GrahamDemo 800 &（とタイプしてその後リターンキーを押す）
として、開かれたパネル上で様々な点数を入力して上記のことを繰り返す。

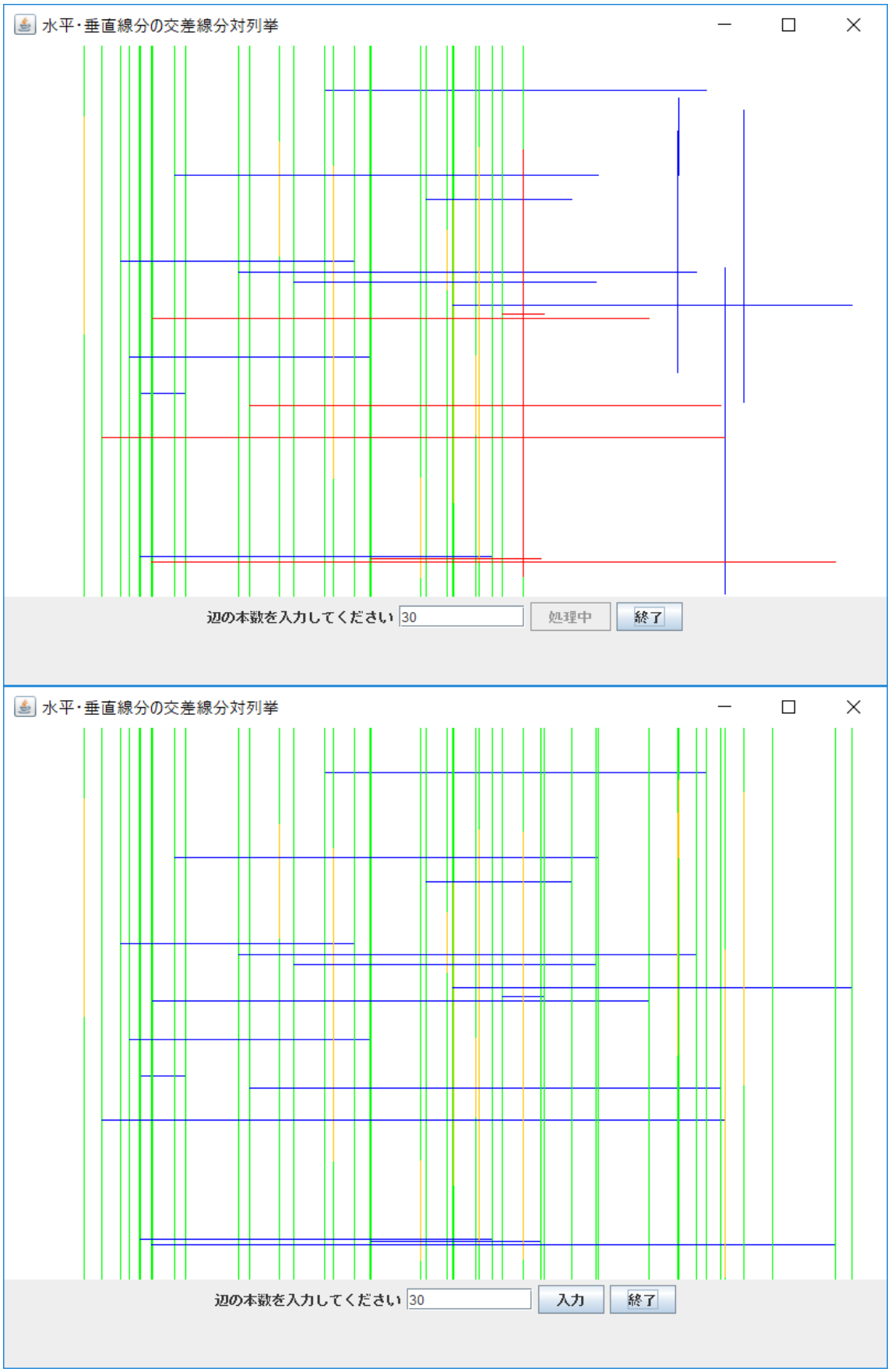
次に、水平・垂直線分の公差線分対列挙について述べる。
e:¥JavaDemoPrograms>
で
java HVIntDemo 800 &（とタイプしてその後リターンキーを押す）
とすると、以下の水平・垂直線分の交差線分対列挙のパネルが開かれる。



ここで、線分数を入力する（線分数を記入して「入力」のボタンを押す)。
すると線分数に等しい水平線分と垂直線分がランダムに発生され、以下のようなパネルの更新が観察できる（線分数は 30 として入力している)。

線分は青で表示される。線分の端点を通る緑の垂直線が走査線であり、最も左の点から順次右に移動していく。垂直線分の端点に走査線が来ると、垂直線分は赤で表示され、それと交差する水平線分が下の方から赤で表示される。なお、交差の様子が目で見て確かめられるように、次のステップに移るまで適切な待ち時間を設定している。


水平・垂直線分の交差線分対列挙
辺の本数を入力してください
30
処理中
終了
水平・垂直線分の交差線分対列挙
辺の本数を入力してください
30
処理中
終了


水平・垂直線分の交差線分対列挙
辺の本数を入力してください
30
処理中
終了
水平・垂直線分の交差線分対列挙
辺の本数を入力してください
30
入力
終了

終了するときには「終了」のボタンを押す。すると、コマンドプロンプトの画面に、再度
e:¥JavaDemoPrograms>
と表示される。繰り返したいときは、そこで再度
java HVIntDemo 800 &（とタイプしてその後リターンキーを押す）
として、開かれたパネル上で様々な線分数を入力して上記のことを繰り返す。