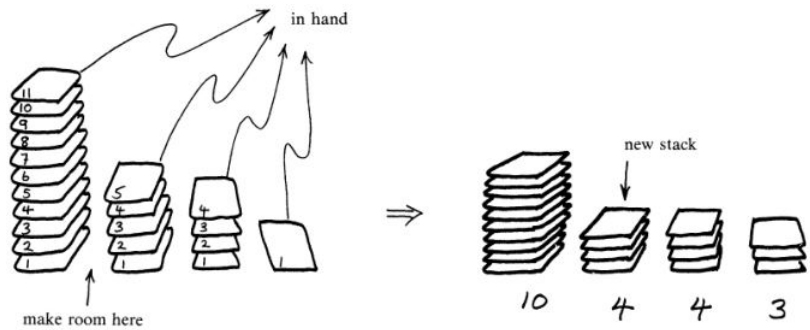
CS 7A - Fall 2019 - Stax. Due 12/10/19

Background Theory



Stax is played with a set of N objects we'll call blox. The first thing you do is to divide the N blox into one or more stax of blox. For example, 21 blox could be divided into four stax consisting of 11, 5, 4, and 1 blox. Now you make a move by taking one blox from each of the stax. Since it doesn't matter which blox you take, you might as well take the top blox. In the example you would get four blox and put these down to form a new stax. This gives you four stax consisting of 10, 4, 4, and 3 blox. If you do this 24 more times you get six stax consisting of 6, 5, 4, 3, 2, 1 blox. This is a steady state position with peroid 1 because it repeats with any subsequent move. If the number of stacks, k, consists of $k, k-1, k-2, \ldots, 3, 2, 1$ blox. This means the process won't reach a steady state unless you start with $N = 1+2+3+\cdots+k = k(k+1)/2$ blox, for some k. These numbers are called triangular numbers, $T_k = \dfrac{k(k+1)}{2}$.

Use pencil and paper to consider various scenarios with $k = 6$ and $T_6 = 21$ to test the following conjecture:

In any arrangement of stax with $\dfrac{k(k+1)}{2}$ blox the procedure reaches a 1-cycle in $\leq k(k-1)$ moves.

The arrangement above took 25 moves to reach a steady state; however the initial division of 21 cards into seven stacks with 5,5,4,3,2,1,1 cards requires 30 moves.

1. Write code to meet these specifications:
   Precondition: A user supplied sequence of numbers in decreasing order (such as 5,5,4,3,2,1,1) which sum to some triangular number, T_n.
   Postcondition: A sequence of Stax moves printed to the console. A typical run with the first line entered by the user and the following line computed by the program is below:

```
Enter a sequence of positive numbers. Enter 0 when done:
1 1 2 3 4 5 5 0
1 2 3 4 4 7
1 2 3 3 6 6
1 2 2 5 5 6
1 1 4 4 5 6
3 3 4 5 6
2 2 3 4 5 5
1 1 2 3 4 4 6
```

```
1 2 3 3 5 7
1 2 2 4 6 6
1 1 3 5 5 6
2 4 4 5 6
1 3 3 4 5 5
2 2 3 4 4 6
1 1 2 3 3 5 6
1 2 2 4 5 7
```

```
1 1 3 4 6 6
2 3 5 5 6
1 2 4 4 5 5
1 3 3 4 4 6
2 2 3 3 5 6
1 1 2 2 4 5 6
1 1 3 4 5 7
2 3 4 6 6
```

```
1 2 3 5 5 5
1 2 4 4 4 6
1 3 3 3 5 6
2 2 2 4 5 6
1 1 1 3 4 5 6
2 3 4 5 7
1 2 3 4 5 6
```

   Here's some starter code that may (or may not) be helpful:

```
//<your name here>
//Working the stax   algorithm

///prototypes
bool endState(vector<int> v);
///Return true only if the input is a vector of the form {1,2,3,...,k}

void show_stacks(vector<int> v);
///Print the vector v

void show_history(vector< vector<int> > history);
```

```
///Print the sequence of vectors in history

bool is_new(vector<int> v, vector< vector<int> > history)
///Compare the vector v (the latest addition to history)
///with all the previous vectors in history.  If there's no match
///return true.  If there is a match, return false

void move(vector<int>& v);
///Perform the Stax move on v
///note that, since v is passed by reference, it will change v.

int main() {
    vector<int> vStax;
    vector< vector<int> > playHistory;
///Get the user's input for the initial vector of stax
///and start building playHistory with the first vector.

///loop the moves, building the playHistory
///and checking at each loop
///for endState(vStax)) and is_new(vStax,playHistory)

///When the a root loop is found (or an end state), display
///the number of moves it took to get there and the size of the root loop.
}
```

2. Now go further to report the number of moves to a loop and the length of the loop. Here are a few typical runs of the code I wrote for this:

```
Enter a sequence of positive numbers. Enter 0 when done:
1 1 2 3 4 5 0

The root loop size is = 6
It takes 6 moves to get to the steady state:

1 1 2 3 4 5
1 2 3 4 6
1 2 3 5 5
1 2 4 4 5
                    1 3 3 4 5
                    2 2 3 4 5
                    1 1 2 3 4 5
==============================================================
Enter a sequence of positive numbers. Enter 0 when done:
7 6 5 4 0

The root loop size is = 7
It takes 26 moves to get to the steady state:

7 6 5 4             2 4 5 5 6           1 2 4 5 5 5
3 4 4 5 6           1 3 4 4 5 5         1 3 4 4 4 6
2 3 3 4 5 5         2 3 3 4 4 6         2 3 3 3 5 6
1 2 2 3 4 4 6       1 2 2 3 3 5 6       1 2 2 2 4 5 6
1 1 2 3 3 5 7       1 1 2 2 4 5 7       1 1 1 3 4 5 7
1 2 2 4 6 7         1 1 3 4 6 7         2 3 4 6 7
1 1 3 5 6 6         2 3 5 6 6           1 2 3 5 5 6
                                                        1 2 4 4 5 6
                                                        1 3 3 4 5 6
                                                        2 2 3 4 5 6
                                                        1 1 2 3 4 5 6
                                                        1 2 3 4 5 7
                                                        1 2 3 4 6 6
```

Submit the code for problems 1 and 2 using your initials in the usual format: {your initials}_Stax.cpp