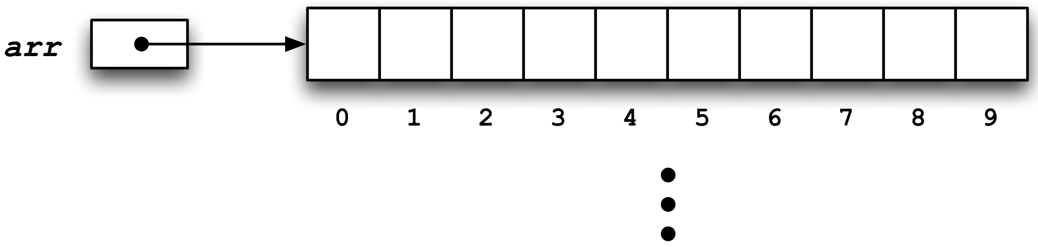
Filling an array from the left:

```
int[] arr  =  new int[10] ;
```

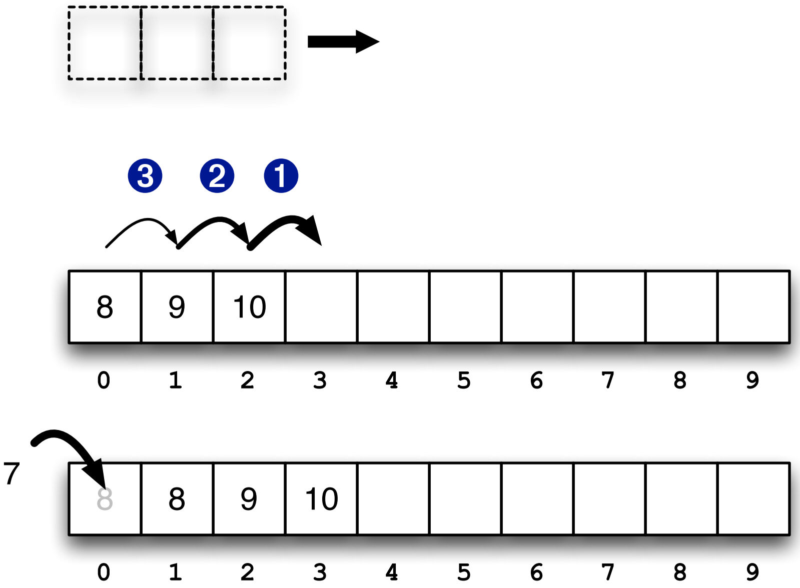


At step i, we shove down a “sub-array” of length i to make room for the new value at position 0. Start shoving from the right, toward the left (steps 1,2,3 below).

writing down these “concrete steps” can help!

```
arr[3] = arr[2];
arr[2] = arr[1];
arr[1] = arr[0];
```

```
arr[0] = 7;
```



Note that we can distinguish:

- **the end result of the filling (“up”, from low to hi, or “down”, from hi to low)**
- **the direction the loop indices [i] actually ran (again, up or down)**
- **the direction in which we filled (from left-to-right or right-to-left)**

For the lab:

- **write a StopWatch class to use for timing (methods: `start`, `stop`, `read`, `clear?`, …);**
- **write a “harness” main class (`TimeTests`) that compares several ways of filling arrays and array lists, with regard to timing;**
- **first: fill arrays and `ArrayLists` from the *left,* using `add(0, _ )` for the `ArrayLists` and using “shoving” for the arrays.**
- **next: fill arrays and `ArrayLists` from the *right,* with `add( _ )` for the `ArrayLists` and plain filling for the arrays (no shoving needed);**
- **all structures should be filled with 1–n, (verified at both ends), no matter the fill direction *(note:* not *0–(n-1), but 1–n)*;**
- **compute timings for size n among the whole powers of 10 between 3 and 6 (i.e., from one thousand to one million, inclusive);**
- **report the various timings in some nice, readable, columnar format (use `printf`): two structures, two directions, four sizes.**
- **Extra fun: append (join together) n copies of some string, using either the `String` class and +, or `StringBuilder`); compare timings for various (big) sizes n.**