The BlockList is an arraydeque of arraydeques. We will call the arraydeque that contains arraydeques the main arraydeque, and the arraydeques contained within the main arraydeque we will call blocks. Let r be the number of blocks, thus they are numbered 0 to r-1. The first (block 0) and last (block r-1) blocks can have anywhere from 1 to b items. That is, they may be partially full. We call these the end blocks. The blocks in locations 1 through r-2 should have exactly b items, never more and never less (unless you are in the process of adding something and shifting items). We will call these inner blocks.

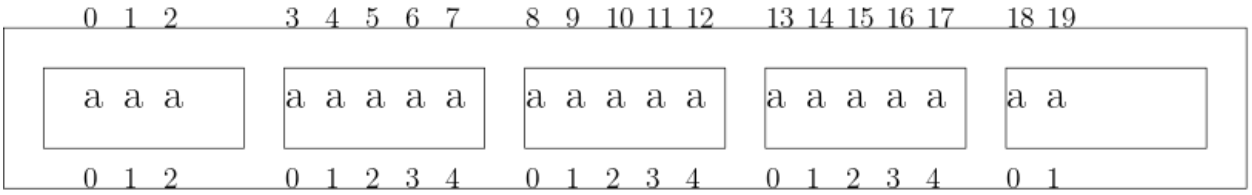
You should never need to call ‘spread()’ or ‘gather()’. This is a different data structure. This data structure need only shift elements.

Let's consider an example, pictured below. The top row of numbers are the indices into the BlockedList. The bottom row of numbers is the corresponding indices into the blocks. I have not included the block indices, but they are 0 through 4 from left to right. I have 5 blocks, and block size is b=5. I call add(9,x). We find the proper block, block 2, and the proper index, 1, and call add(1,x) on block 2. Since the inner blocks all have b items, we can quickly find the proper block and index using some simple math.

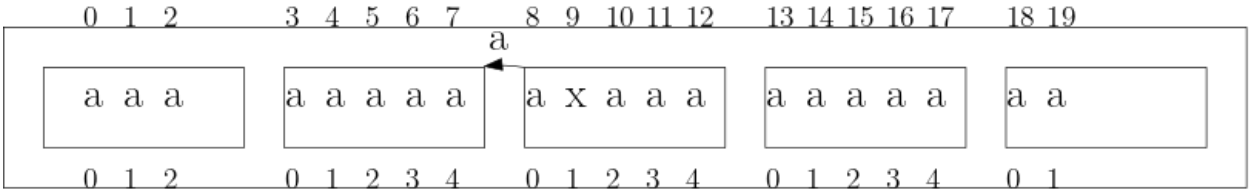
Since block 2 is an inner block, it has precisely 5 items. Since we have added an item, it now has 6, which is too many. We calculate the direction to shift that requires the least work. In this case both directions require 2 shifts, so either works.

We shift an element from block 2 to block 1. Block 1 now has too many items. We shift an element from block 1 to block 0. Block 0 has less than b elements, but it is an end block, so that is ok. If block 0 had b items when we shifted, we would add a new, empty block at location 0, which becomes the new end block, and shift an element into it.

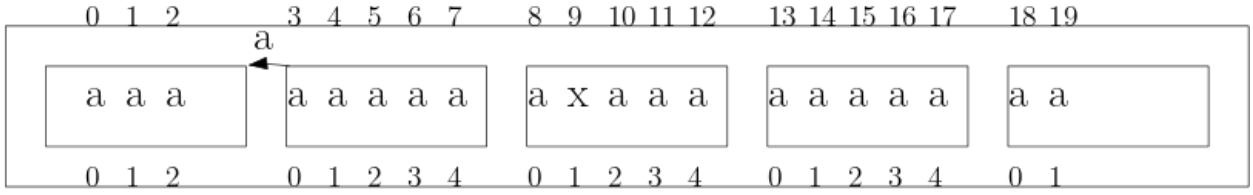
Consider a call to remove(9), not pictured, but essentially the reverse of the add operation. After removing element 9, block 2 would have less than b items. We compute the direction that requires the least shifts, and shift an element from block 1 to block 2. Now block 1 has too few items, so we shift an item from block 0 to block 1. Block 0 has less than b items which is ok. If block 0 was empty, we would remove that block.



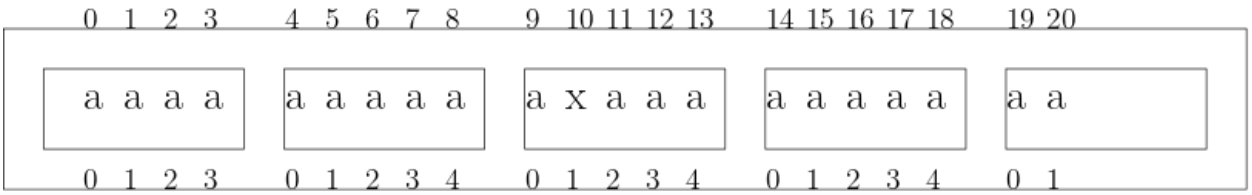
add(9, x)



Block 2 has more than b items. Shift an 'a' from block 2 to block 1.



Block 1 has more than b items. Shift an 'a' from block 1 to block 0.



Block 0 has less than b items, which is ok since it is the first block.