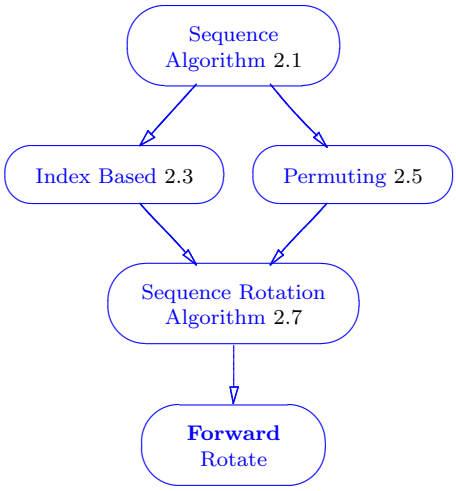
2.7.1 Forward Rotate

Section authors: Cho Yeung, Haiying Ren, Mayuresh Kulkarni.



Refinement of: Sequence Rotation Algorithm (§2.7), Index Based (§2.3), Permuting (§2.5), Sequence Algorithm (§2.1).

Prototype:

```
template⟨class ForwardIterator⟩
inline ForwardIterator rotate
                (ForwardIterator first,
                 ForwardIterator middle,
                 ForwardIterator last)
```

Effects: Standard effects of a Sequence Rotation Algorithm (§2.7). In brief: the range `[first, middle)` is rotated with the range `[middle,last)`, so that the `middle` element becomes the new first element. This algorithm is used only if the iterators are only forward iterators.

Asymptotic complexity: Let $N = \text{last} - \text{first}$.

- Running Time: $\Theta(N)$

Complexity in terms of operation counts:

- Let $N =$ size of input sequence
- $M =$ position of the middle iterator.
- Formulas:

M Pos.	Val. Asi.	Itr. Ops.	Itr. Asi.
$.1N$	$2.7N$	$1.8N$	$3.6N$
$.5N$	$1.5N$	N	$2N$
$.9N$	$2.7N$	$2.6N$	$3.6N$

- $M = 0.1N$

Input Size	Value Asmts	Iterator Ops	Itr. Asmts
1000	2700	1803	3619
2000	5400	3603	7219
4000	10800	7203	14419
8000	21600	14403	28819
15000	40500	27003	54019
30000	81000	54003	108019

- $M = 0.5N$

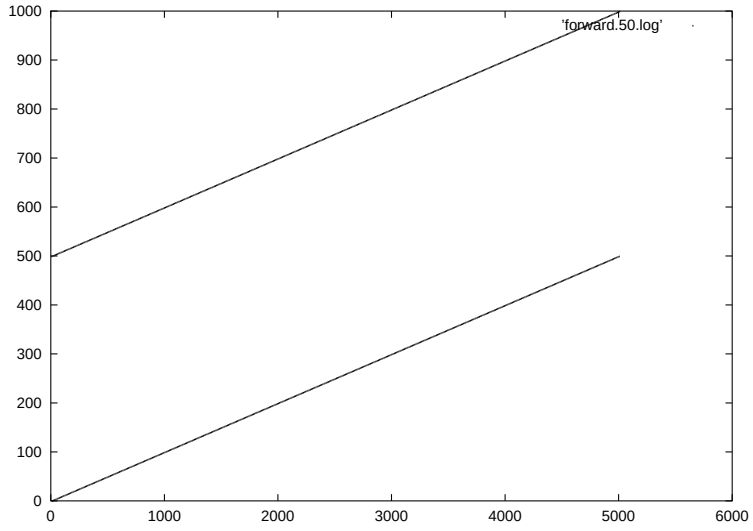
Input Size	Value Asmts	Iterator Ops	Itr. Asmts
1000	1500	1003	2011
2000	3000	2003	4011
4000	6000	4003	8011
10000	15000	10003	20011
30000	45000	30003	60011

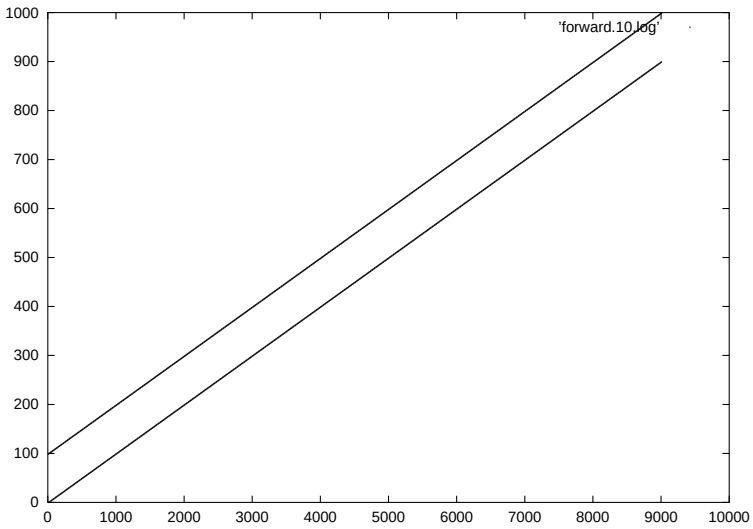
- $M = 0.9N$

Input Size	Value Asmts	Iterator Ops	Itr. Asmts
1000	2700	2602	3618
2000	5400	5202	7218
4000	10800	10402	14418
10000	27000	26002	36018
30000	81000	78002	108018

- This algorithm has two parts. In the first part, it swaps mid with first, mid + 1 with second and so on. Thus,at the end of this part, (mid,last] range has now been transferred to its appropriate position. But, though the elements of the original (first,mid] range are now in their correct positions, they are in a cyclic permutation of the required order. In the second part, a series of swaps is done to rectify this.

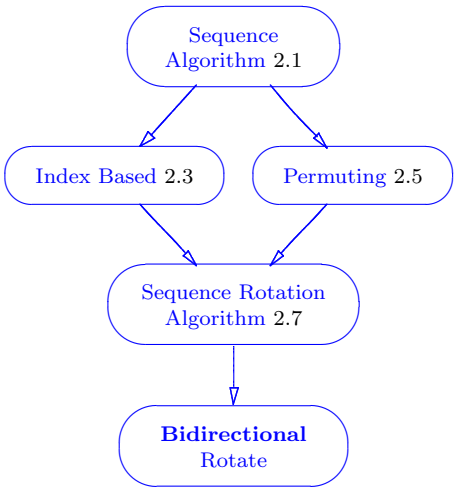
Iterator trace plot:




'forward.10.log'
1000
900
800
700
600
500
400
300
200
100
0
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000

2.7.2 Bidirectional Rotate

Section authors: Cho Yeung, Haiying Ren, Mayuresh Kulkarni.



Refinement of: Sequence Rotation Algorithm (§2.7), Index Based (§2.3), Permuting (§2.5), Sequence Algorithm (§2.1).

Prototype:

```
template⟨class BidirectionalIterator⟩
inline BidirectIonalIterator rotate
               (BidirectionalIterator first,
                BidirectionalIterator middle,
                BidirectionalIterator last)
```

Effects: Standard effects of a Sequence Rotation Algorithm (§2.7). In brief: the range `[first, middle)` is rotated with the range `[middle, last)`, so that the `middle` element becomes the new first element. This algorithm is used only if the iterators are only bidirectional iterators.

Asymptotic complexity: Let $N = \text{last} - \text{first}$.

- Running Time: $\Theta(N)$

Complexity in terms of operation counts:

- Let $N =$ size of input sequence

- $M =$ position of the middle iterator.
- Formulas:

M Pos.	Val. Asi.	Itr. Ops.	Itr. Asi.
$.1N$	$3N$	$2N$	$4.7N$
$.5N$	$3N$	$2N$	$3.5N$
$.9N$	$3N$	$2N$	$4.7N$

Input Size	Value Asmts	Iterator Ops	Itr. Asmts
1000	3000	2007	4713
2000	6000	4007	9413
4000	12000	8007	18813
8000	24000	16007	37613
15000	45000	30007	70513
30000	90000	60007	141013

- *middle* $= N \cdot 0.5$

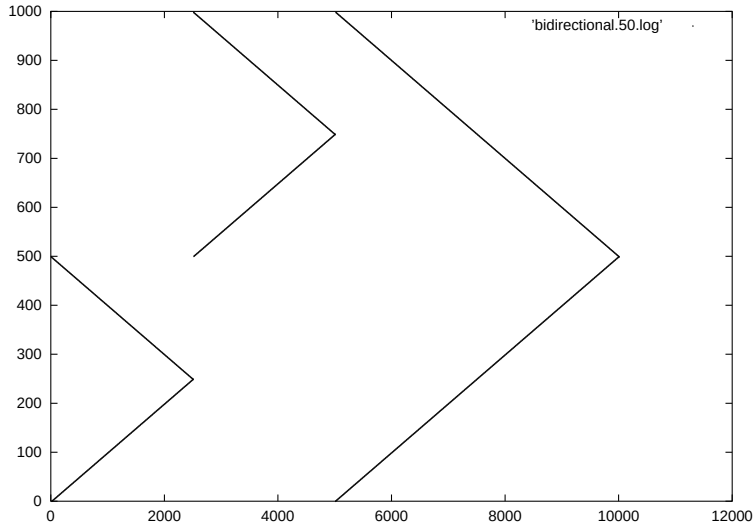
Input Size	Value Asmts	Iterator Ops	Itr. Asmts
1000	3000	2007	3513
2000	6000	4007	7013
4000	12000	8007	14013
10000	30000	20007	35013
30000	90000	60007	105013

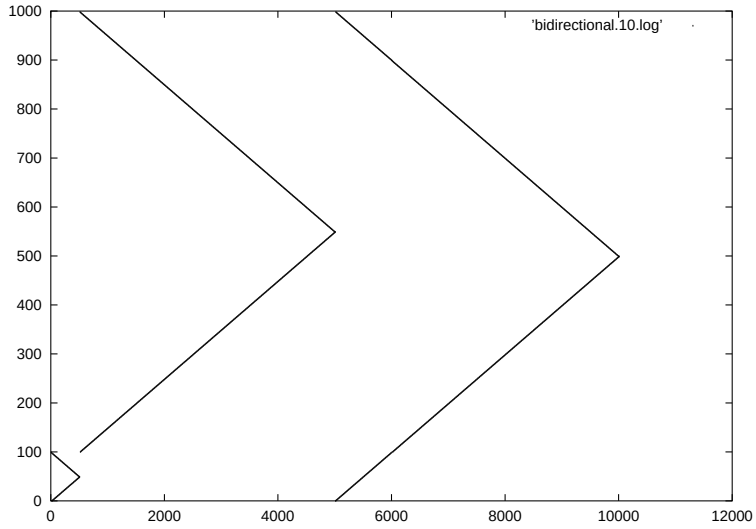
- *middle* $= N \cdot 0.9$

Input Size	Value Asmts	Iterator Ops	Itr. Asmts
1000	3000	2008	4713
2000	6000	4008	9413
4000	12000	8008	18813
10000	30000	20008	47013
30000	90000	60008	141013

- This algorithm first reverses the (first,mid] range. Then it reverses the (mid,last] range. Finally it reverses (first,last] range.

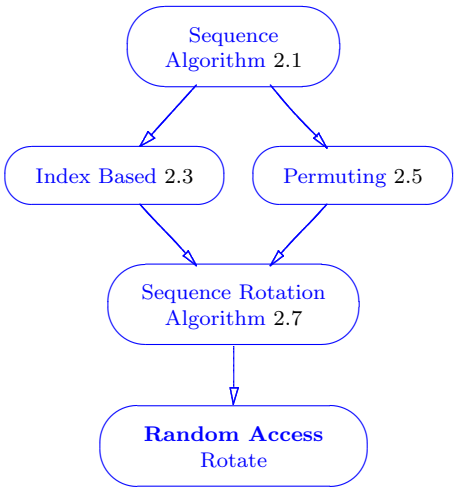
Iterator trace plot:





2.7.3 Random Access Rotate

Section authors: Cho Yeung, Haiying Ren, Mayuresh Kulkarni.



Refinement of: Sequence Rotation Algorithm (§2.7), Index Based (§2.3), Permuting (§2.5), Sequence Algorithm (§2.1).

Prototype:

```
template⟨class RandomAccessIterator⟩
inline RandomAccessIterator rotate
               (RandomAccessIterator first,
                RandomAccessIterator middle,
                RandomAccessIterator last)
```

Effects: Standard effects of a Sequence Rotation Algorithm (§2.7). In brief: the range `[first, middle)` is rotated with the range `[middle, last)`, so that the `middle` element becomes the new first element. This algorithm is used only if the iterators are only random access iterators.

Asymptotic complexity: Let $N = \text{last} - \text{first}$.

- Running Time: $\Theta(N)$

Complexity in terms of operation counts:

- Let $N =$ size of input sequence

- $M =$ position of the middle iterator.
- formulas

M Pos.	Val. Asi.	Itr. Ops.	Itr. Asi.
$.1N$	$1.1N$	$0.9N$	$1.9N$
$.5N$	$1.5N$	$0.5N$	$2N$
$.9N$	$1.1N$	$0.8N$	$1.8N$

- $M = 0.1N$

Input Size	Value Asmts	Iterator Ops	Itr. Asmts
1000	1100	900	1908
2000	2200	1800	3808
4000	4400	3600	7608
8000	8800	7200	15208
15000	16500	13500	28508
30000	33000	27000	57008

- $M = 0.5N$

Input Size	Value Asmts	Iterator Ops	Itr. Asmts
1000	1500	501	2012
2000	3000	1001	4012
4000	6000	2001	8012
10000	15000	5001	20012
30000	45000	15001	60012

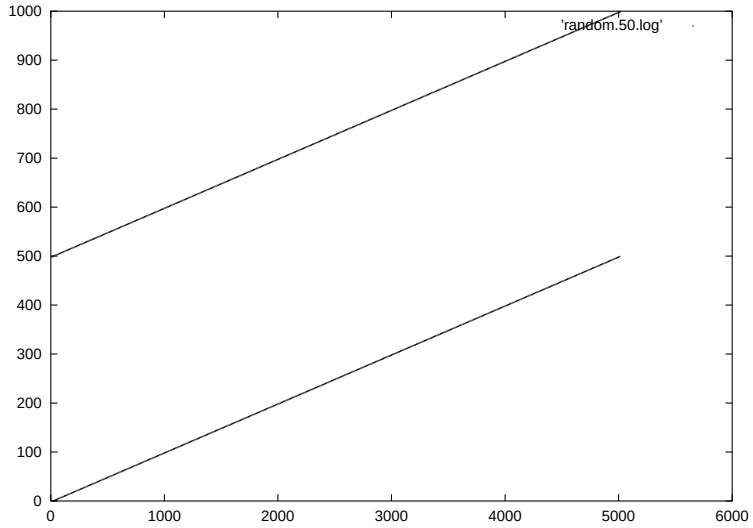
- $M = 0.1N$

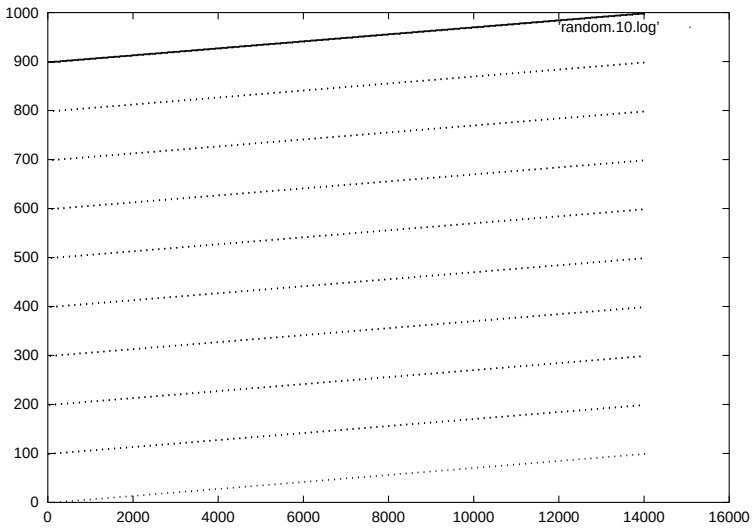
Input Size	Value Asmts	Iterator Ops	Itr. Asmts
1000	1100	800	1808
2000	2200	1600	3608
4000	4400	3200	7208
10000	11000	8000	18008
30000	33000	24000	54008

- The random access rotate algorithm is based on the cycle structure of the permutation of $0 \ldots n-1$ the algorithm performs. The number

of cycles is $g = \gcd(N, M)$ and each cycle is of length $k = N/g$. Within each cycle $k + 1$ value assignments are done: 1 to copy the first element to a temporary location, $k - 1$ to copy the other $k - 1$ elements to their new locations, and 1 to copy from the temporary to the last location. Thus overall $g((N/g) + 1) = N + g$ value assignments are done. The best case, $N + 1$, occurs when $g = 1$, as for example in a rotation of any sequence by 1 element or of any rotation of a sequence with a prime number of elements. The worst case occurs when $g = N/2$. (The algorithm checks for $M = 0$ or N and does nothing in those cases).

Iterator trace plot:




'random.10.log'
1000
900
800
700
600
500
400
300
200
100
0
0
2000
4000
6000
8000
10000
12000
14000
16000