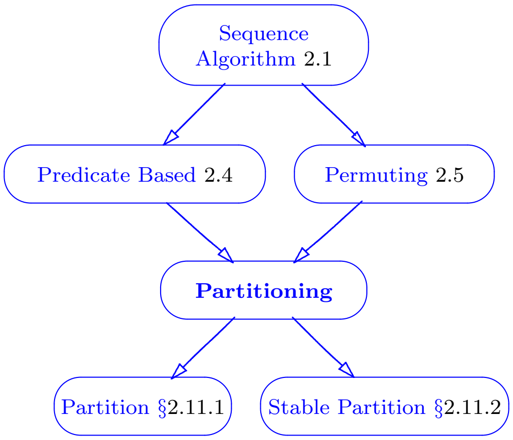
2.11 Partitioning

Section authors: Jin Kyu Gahm, MyungYul Jang, and Lei Deng



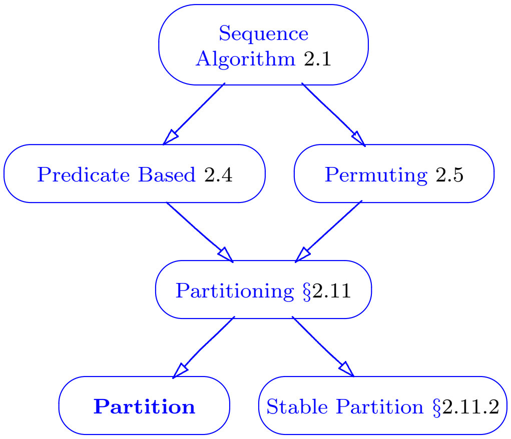
Refinement of: Predicate Based (§2.4), Permuting (§2.5) and Sequence Algorithm (§2.1).

Input: Iterators first and last delimiting a range of elements [first, last), and predicate pred applied to values of the elements.

Output: Predicate middle and a modified sequence of elements in the same range.

Effects: The elements in the range [first, last) after execution are reordered based on predicate pred such that the elements that satisfy pred precede the elements that fail to satisfy it. The postcondition is that, for some iterator middlein the range [first, last), pred(∗i) is true for every iterator i in the range [first, middle) and false for every iterator i in the range [middle, last) (1).

2.11.1 Partition



Refinement of: Partitioning (§2.11), and therefore of Predicate Based (§2.4), Permuting (§2.5), Sequence Algorithm (§2.1).

Prototype: `template` ⟨*class ForwardIterator,* *class Predicate*⟩

```
ForwardIterator partition(
          ForwardIterator first,
          ForwardIterator last,
          Predicate pred)
```

Effects: The elements in the range [first, last) after execution are reordered based on predicate pred such that the elements that satisfy pred precede the elements that fail to satisfy it. The relative order of the elements in both the range [first, middle) and [middle, last) may be different from the original one after execution.

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

- Average case (random data): $O(N)$
- Worst case: $O(N)$

Complexity in terms of operation counts:

Table 1: Performance of Partition on random sequences with different structures(sizes and operations counts in multiples of 1,000)

Size	Sequence type	Comparisons	Assignments	Iterator Ops	Integer Ops	Total Ops
1	Random(Vector)	1	0.622	4.688	0	6.31
	Bidirectional(List)	1	0.49	4.33	0	5.82
	Forward(Slist)	1	0.745	3.748	0	5.493
4	Random(Vector)	4	2.35	19.698	0	26.048
	Bidirectional(List)	4	2.878	19.697	0	26.575
	Forward(Slist)	4	6.355	18.358	0	28.713
16	Random(Vector)	16	11.422	78.954	0	106.376
	Bidirectional(List)	16	8.722	72.509	0	97.231
	Forward(Slist)	16	34.792	82.795	0	133.587
64	Random(Vector)	64	46.951	317.461	0	428.412
	Bidirectional(List)	64	32.113	278.37	0	374.483
	Forward(Slist)	64	58.774	250.771	0	373.539
256	Random(Vector)	256	96.457	1047.91	0	1400.36
	Bidirectional(List)	256	109.96	1062.2	0	1428.16
	Forward(Slist)	256	170.386	938.389	0	1364.78
1024	Random(Vector)	1024	564.394	4600.49	0	6188.88
	Bidirectional(List)	1024	694.615	5058.73	0	6777.34
	Forward(Slist)	1024	1521.31	4593.31	0	7138.62

- When the type of input is vector

  Value Comparisons: (§A.1) N

  Value Assignments: (§A.2.1) $0.55N - 4.0$

  Iterator Operations: (§A.2.2) $4.5N - 7.8$

  Total Operations: (§A.2.3) $6N - 11.8$

- When the type of input is list

  Value Comparisons: (§A.1) N

  Value Assignments: (§A.3.1) $0.68N - 13$

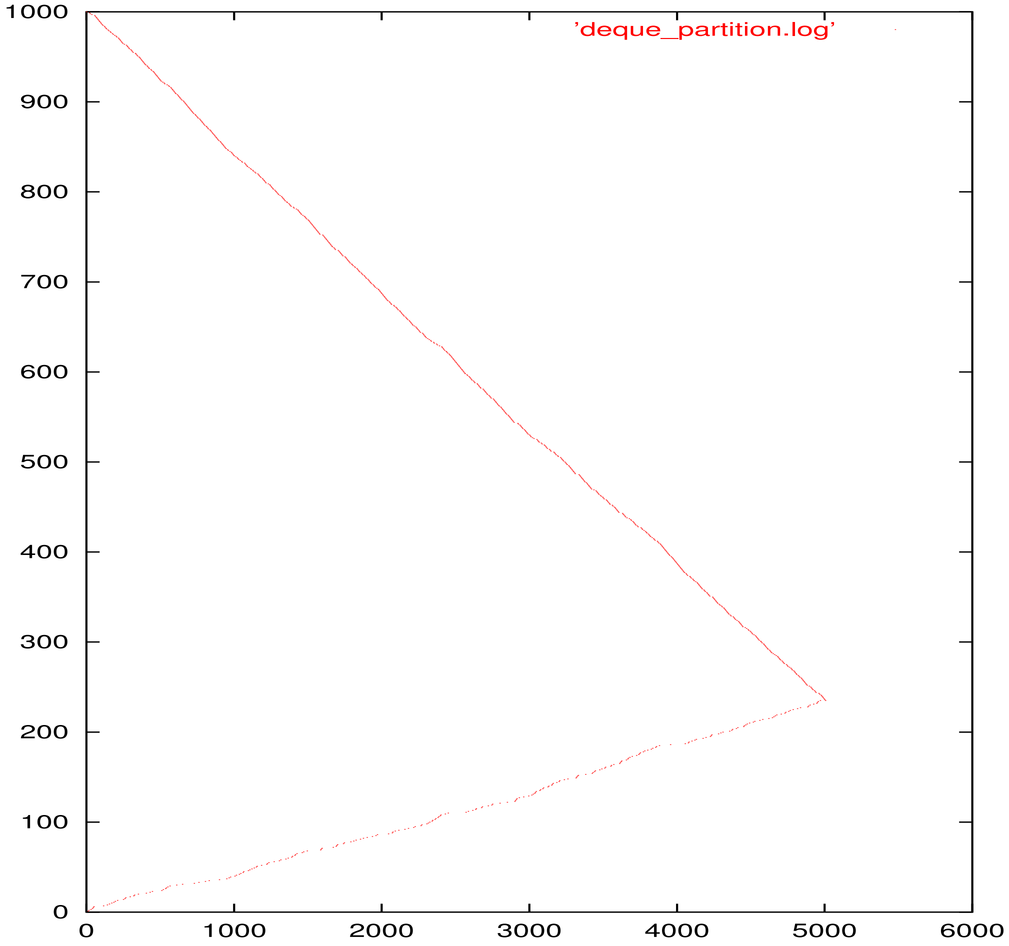
  Iterator Operations: (§A.3.2) $4.9N - 41.8$

  Total Operations: (§A.3.3) $6.6N - 54.8$

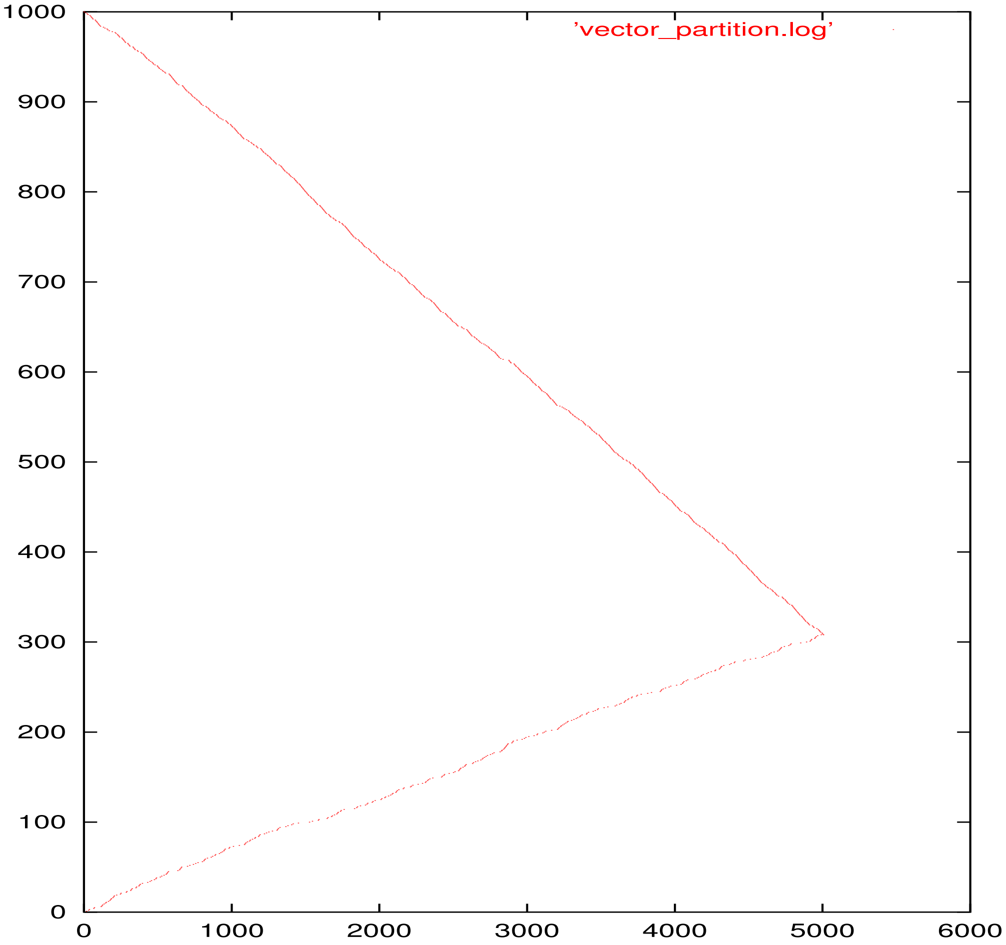
- When the type of input is slist

Value Comparisons: (§A.1)	N
Value Assignments: (§A.4.1)	$0.4N \lg N - 2.6N + 29$
Iterator Operations: (§A.4.2)	$4.5N - 38.6$
Total Operations: (§A.4.3)	$7.0N - 77.2$

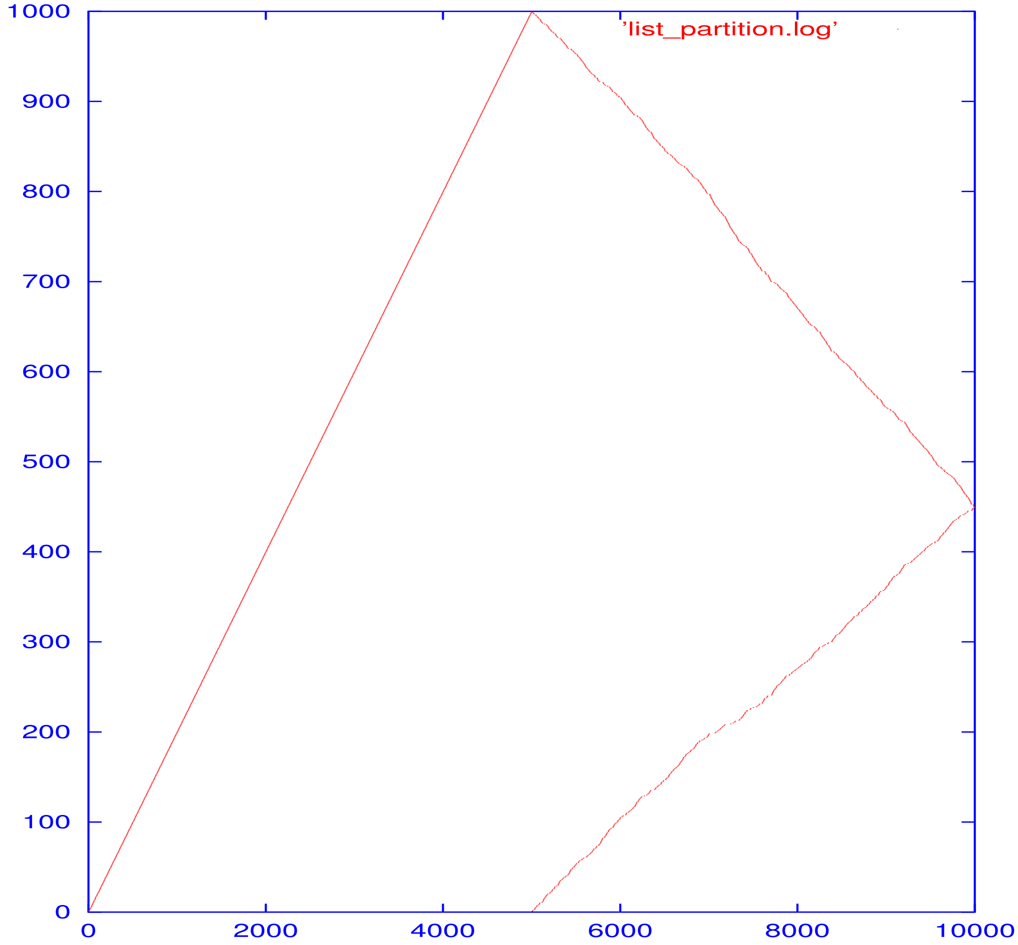
Iterators trace plots:



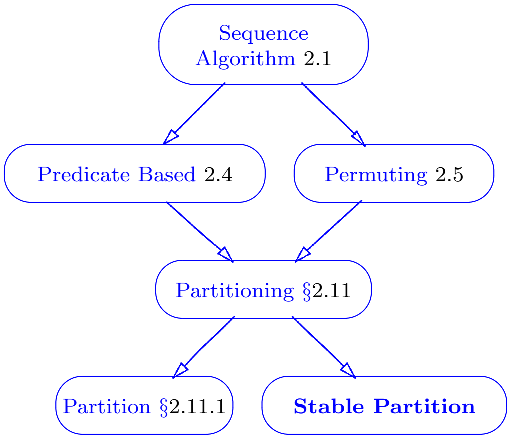
This scans from both front and end. If an element that violates predicate condition is found from one iteration, that iteration stops then find the violated value in the other iteration. Then swap two values, then scan rest elements.



Since vector uses bidirectional, it's similar to deque. This scans from both front and end. If an element that violates predicate condition is found from one iteration, that iteration stops then find the violated value in the other iteration. Then swap two values, then scan rest elements.


1000
900
800
700
600
500
400
300
200
100
0
0
2000
4000
6000
8000
10000
'list_partition.log'

2.11.2 Stable Partition



Refinement of: Partitioning (§2.11), and therefore of Predicate Based (§2.4), Permuting (§2.5), Sequence Algorithm (§2.1).

Prototype: `template` ⟨*class ForwardIterator,* *class Predicate*⟩

```
ForwardIterator stable_partition(
          ForwardIterator first,
          ForwardIterator last,
          Predicate pred)
```

Effects: The elements in the range [first, last) after execution are reordered based on predicate pred such that the elements that satisfy pred precede the elements that fail to satisfy it. It is noted that is that the relative order of the elements in both the range [first, middle) and [middle, last) is preserved after execution.

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

- Average case (random data with buffer): $O(N)$
- Worst case (without buffer, `in_place_stable_partition`): $O(N \lg N)$

If the available memory for a buffer is smaller than the range [first, last), then the `stable_partition` function requires $O(N \lg N)$ time and perform $N \lg N$ swaps, where N is the size of the range [first, last). If there is enough available memory for the buffer to contain all of the elements in the range [first, last), then the `stable_partition` function requires linear time, performing $N+m$ assignments operations and applying the predicate exactly N times, where m is the size of the range [middle, last). If there is not enough memory, the function is recursively called on halves of the sequence, making each half stably partitioned and then performs rotation to create a final range of [first, last) that is stably partitioned.

Complexity in terms of operation counts:

- When the type of input is vector

Value Comparisons: (§A.1)	N
Value Assignments: (§A.5.1)	$2.5N + 12.8$
Iterator Operations: (§A.5.2)	$6N$
Total Operations: (§A.5.3)	$9.5N + 12.8$

- When the type of input is list

Value Comparisons: (§A.1)	N
Value Assignments: (§A.6.1)	$2.6N + 5.7$
Iterator Operations: (§A.6.2)	$8N$
Total Operations: (§A.6.3)	$11.6N + 5.7$

- When the type of input is slist

Value Comparisons: (§A.1)	N
Value Assignments: (§A.7.1)	$2.6N + 5.7$
Iterator Operations: (§A.7.2)	$8N$
Total Operations: (§A.7.3)	$11.6N + 5.8$

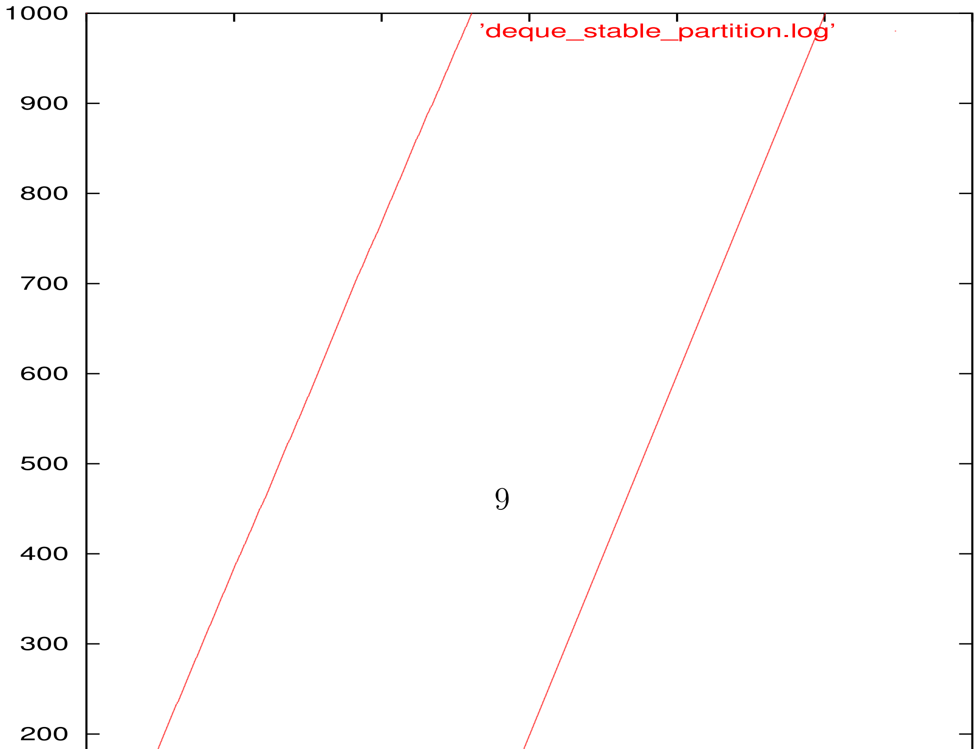
Worst case operation counts: When the buffer was disabled and only the `in_place_stable_partition` function was called, the worst case happens.

Value Comparisons: (§A.1)	N
Value Assignments: (§A.8.1)	$0.5N \lg N + 9.4N + 0.4$
Iterator Operations: (§A.8.2)	$2.6N \lg N + 63.9N - 100.3$
Total Operations: (§A.8.3)	$1.2N \lg N + 173N - 590$

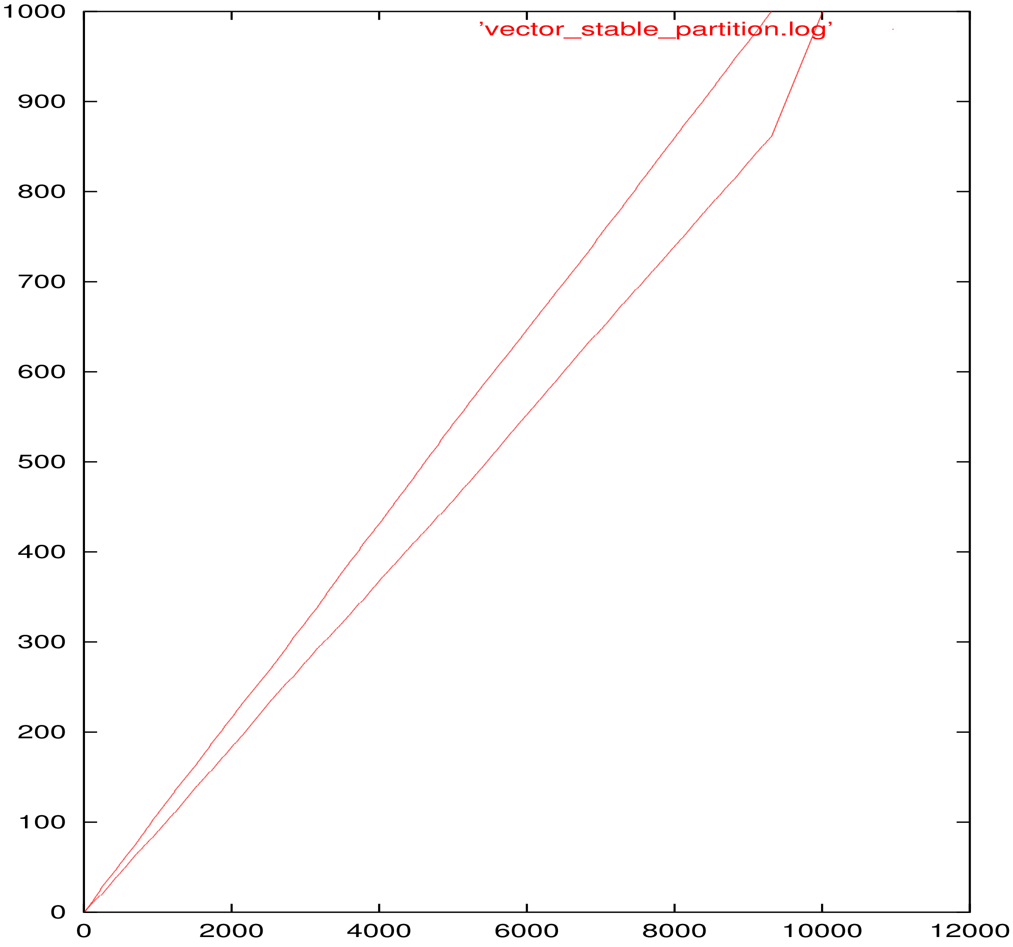
Iterator trace plots:

Table 2: Performance of Stable Partition on random sequences with different structures(sizes and operations counts in multiples of 1,000)

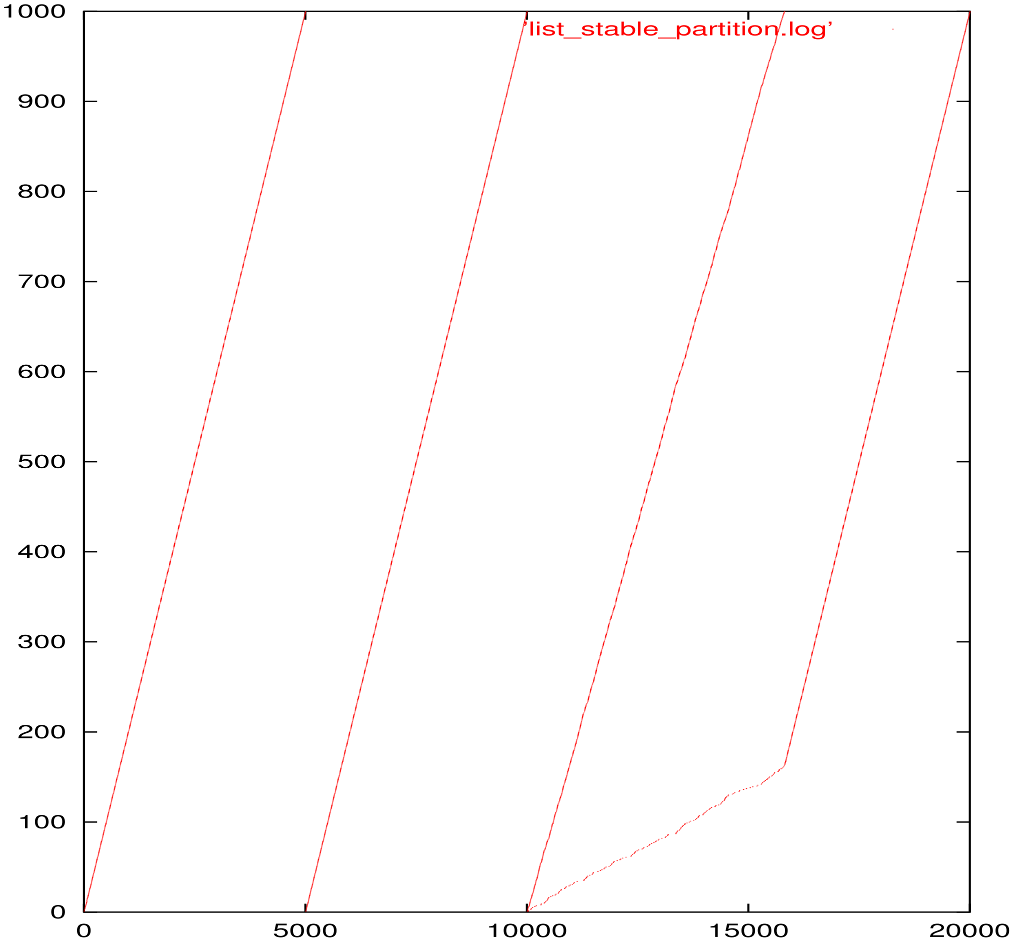
Size	Sequence type	Comp- arisons	Assign- ments	Iterator Ops	Integer Ops	Total Ops
1	Random(Vector)	1	2.434	6.024	0.005	9.463
	Bidirectional(List)	1	2.786	8.024	0.003	11.813
	Forward(Slist)	1	2.786	8.024	0.003	11.813
4	Random(Vector)	4	10.553	24.024	0.005	38.582
	Bidirectional(List)	4	9.915	32.024	0.003	45.942
	Forward(Slist)	4	9.915	32.024	0.003	45.942
16	Random(Vector)	16	39.706	96.024	0.005	151.735
	Bidirectional(List)	16	37.664	128.024	0.003	181.691
	Forward(Slist)	16	37.664	128.024	0.003	181.691
64	Random(Vector)	64	159.317	384.024	0.005	607.346
	Bidirectional(List)	64	173.157	512.024	0.003	749.184
	Forward(Slist)	64	173.157	512.024	0.003	749.184
256	Random(Vector)	256	706.576	1536.02	0.005	2498.61
	Bidirectional(List)	256	712.186	2048.02	0.003	3016.21
	Forward(Slist)	256	712.208	2048.02	0.003	3016.21
1024	Random(Vector)	1024	2519.19	6144.02	0.005	9687.22
	Bidirectional(List)	1024	2699.33	8192.02	0.003	11915.4
	Forward(Slist)	1024	2699.33	8192.02	0.003	11915.4



The left line scans the container. If a value violate the predicate condition, put it into the buffer. This is shown by the middle line. The right line is just to copy the buffer into the container. If there is nothing to split the container somehow by the condition(eg. pivot), the middle would not be shown up and the left and right would be put together as one.

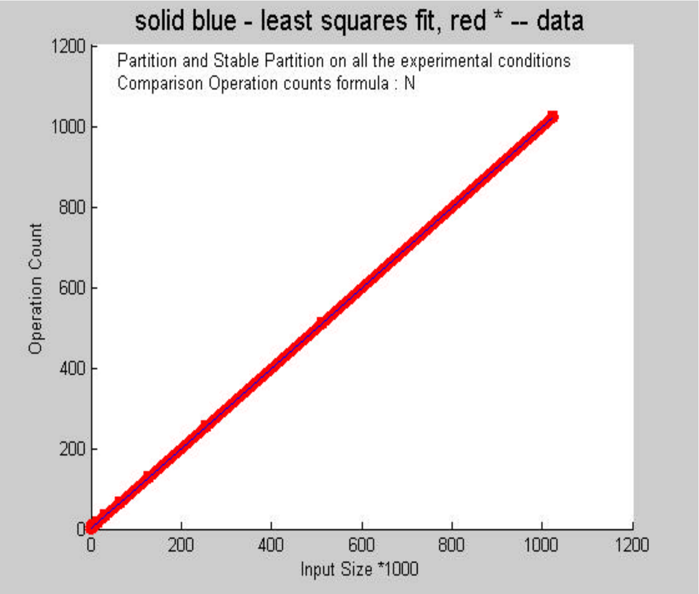


Since vector uses bidirectional, it's similar to deque. The left line scans the container. If a value violate the predicate condition, put it into the buffer. This is shown by the middle line. The right line is just to copy the buffer into the container. If there is nothing to split the container somehow by the condition(ex, pivot), the middle would not be shown up and the left and right would be put together as one.


'list_stable_partition.log'
1000
900
800
700
600
500
400
300
200
100
0
0
5000
10000
15000
20000

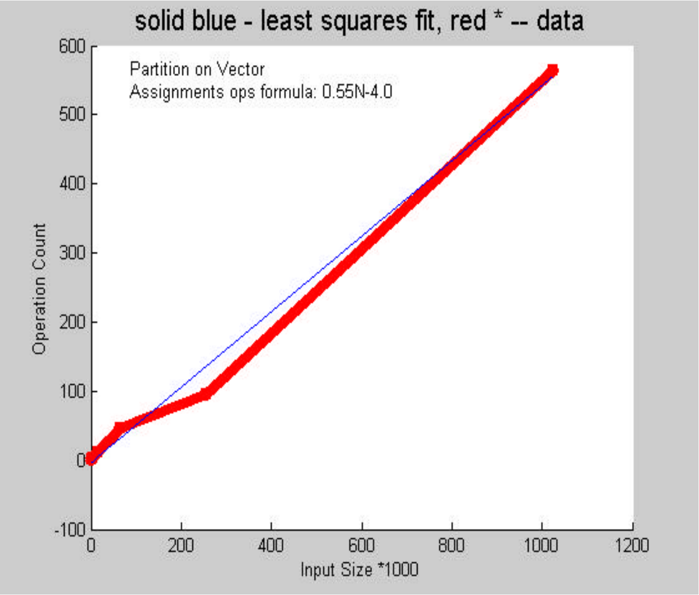
A Curve fitting to measured counts

A.1 Value Comparisons

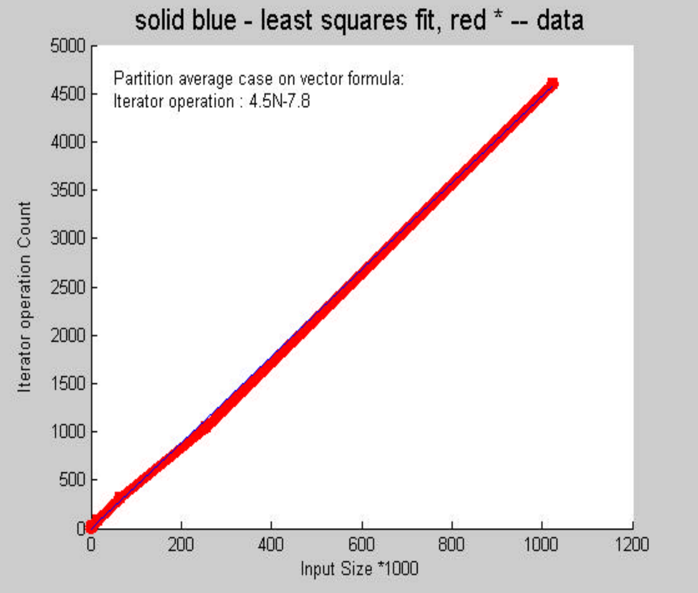


A.2 Vector - Partition

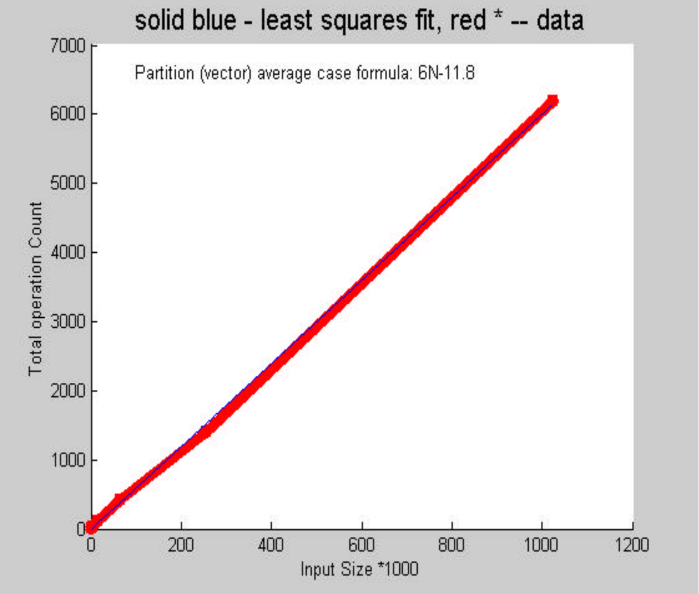
A.2.1 Value Assignments



A.2.2 Iterator Operations

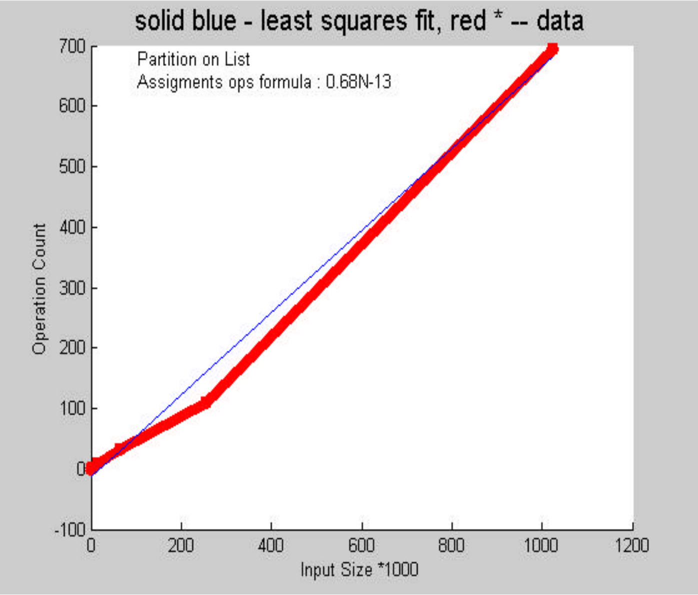


A.2.3 Total Operations

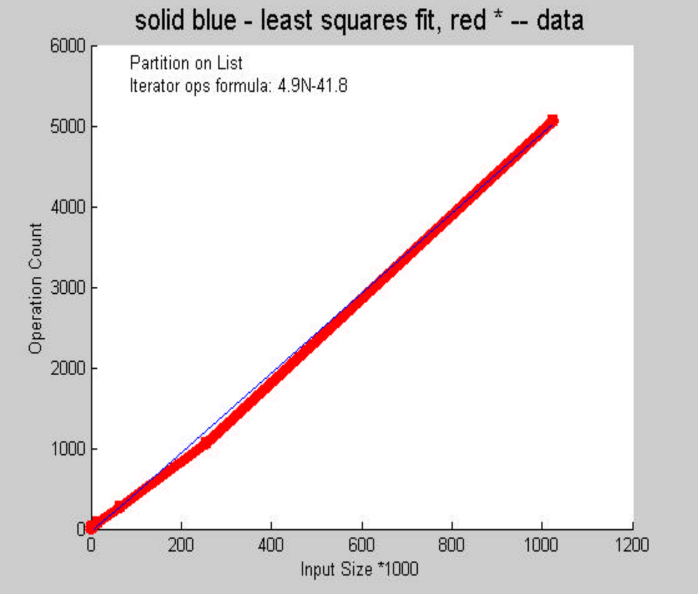


A.3 List - Partition

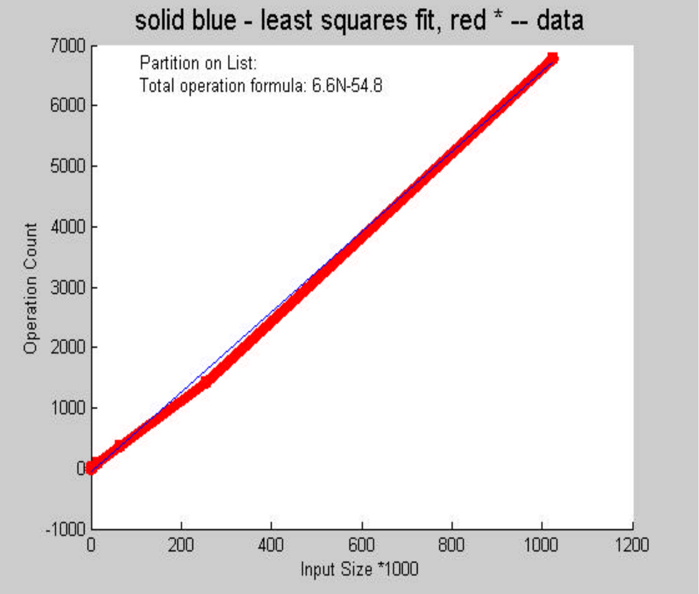
A.3.1 Value Assignments



A.3.2 Iterator Operations

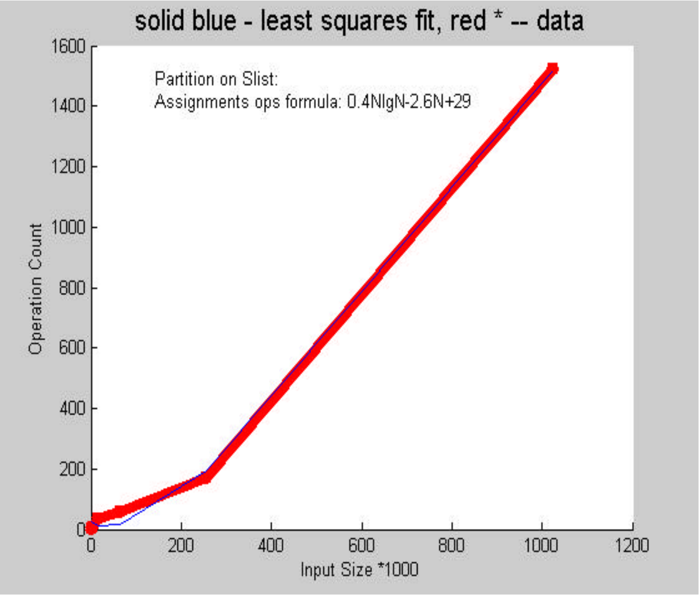


A.3.3 Total Operations

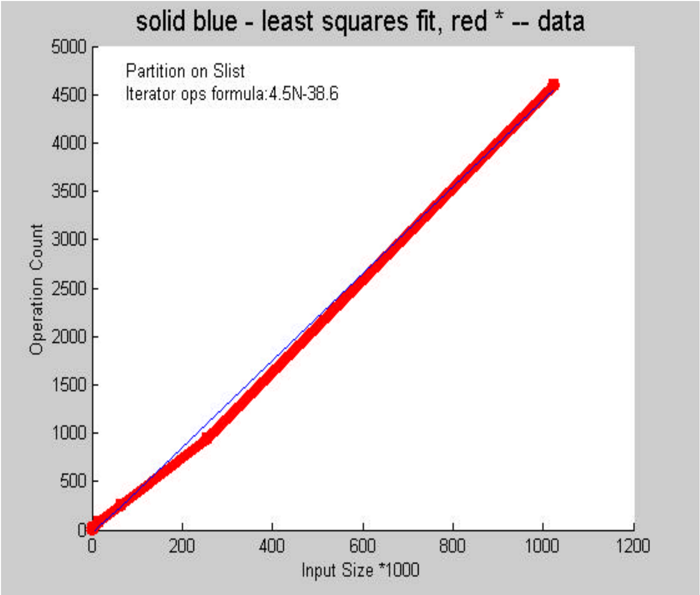


A.4 Slist - Partition

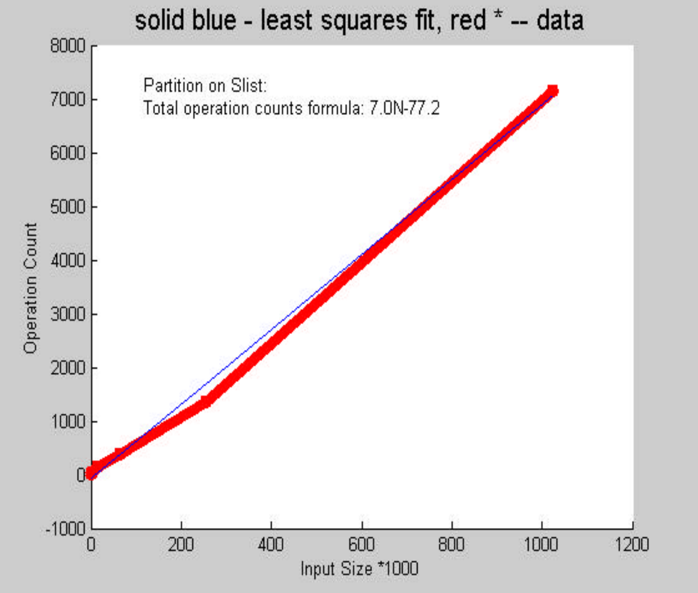
A.4.1 Value Assignments



A.4.2 Iterator Operations

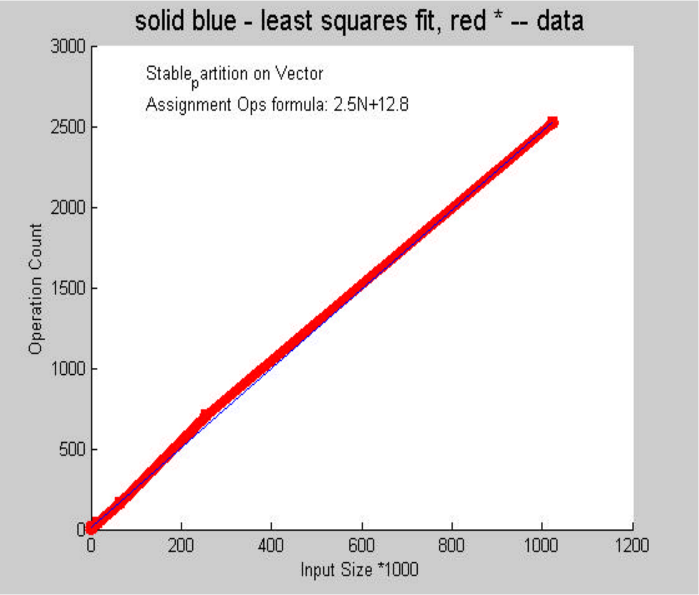


A.4.3 Total Operations

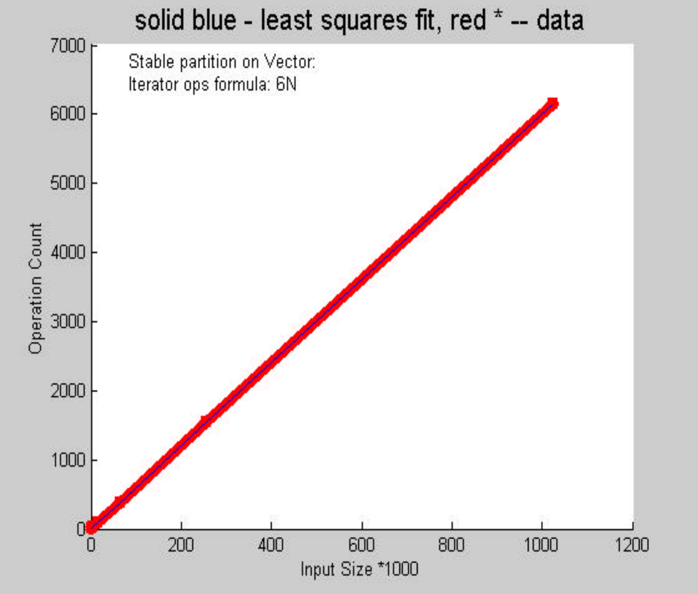


A.5 Vector - Stable Partition

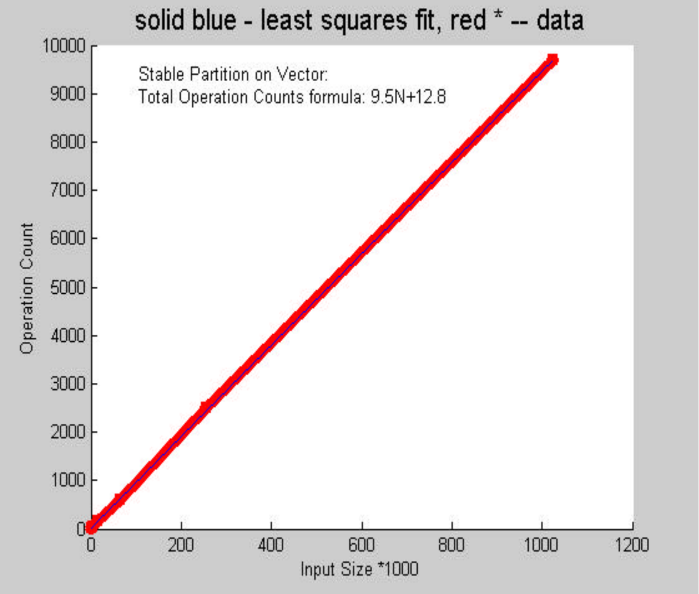
A.5.1 Value Assignments



A.5.2 Iterator Operations

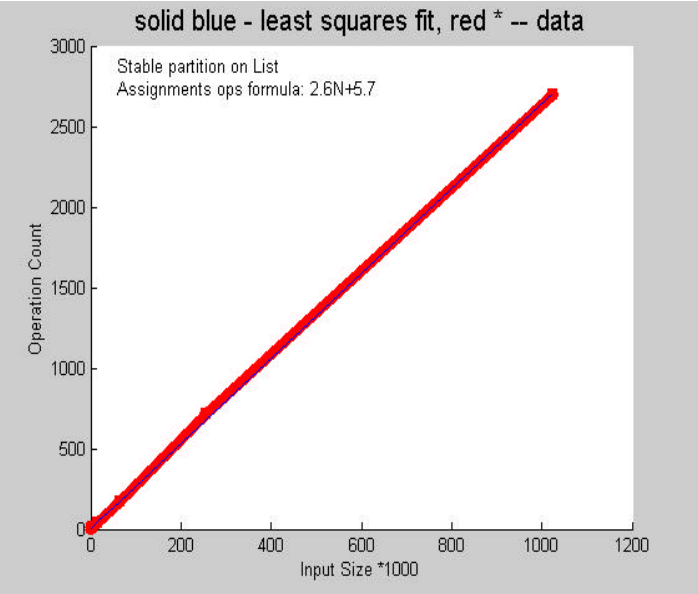


A.5.3 Total Operations

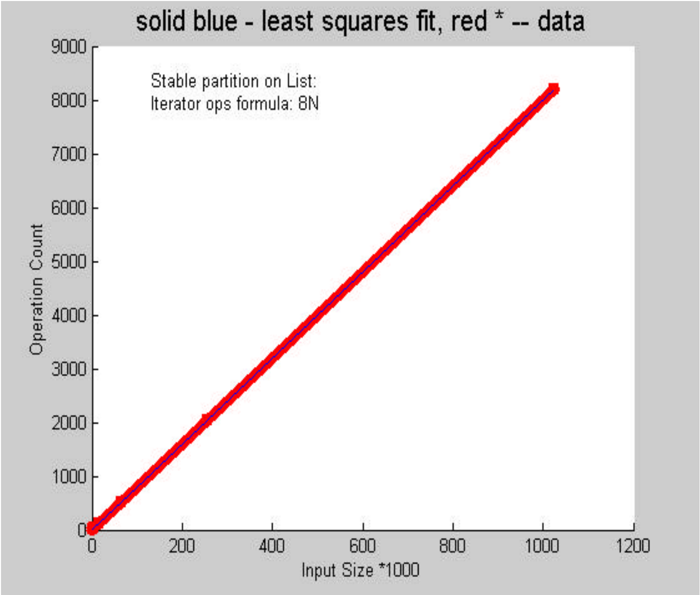


A.6 List - Stable Partition

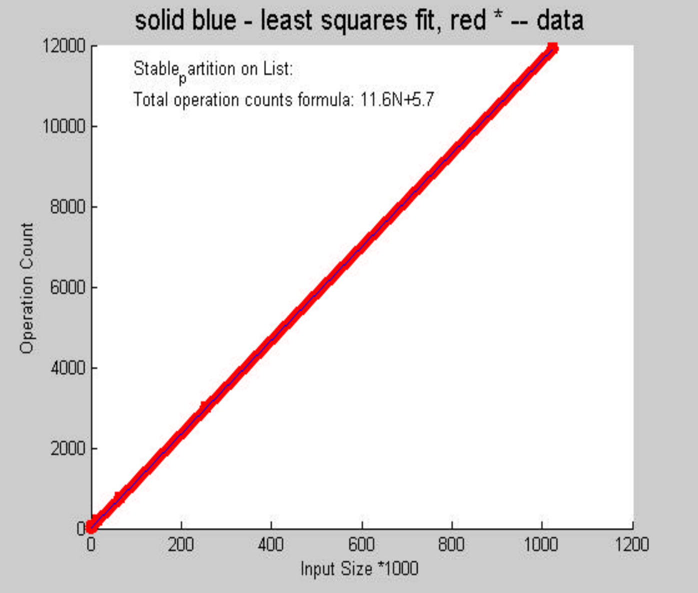
A.6.1 Value Assignments



A.6.2 Iterator Operations

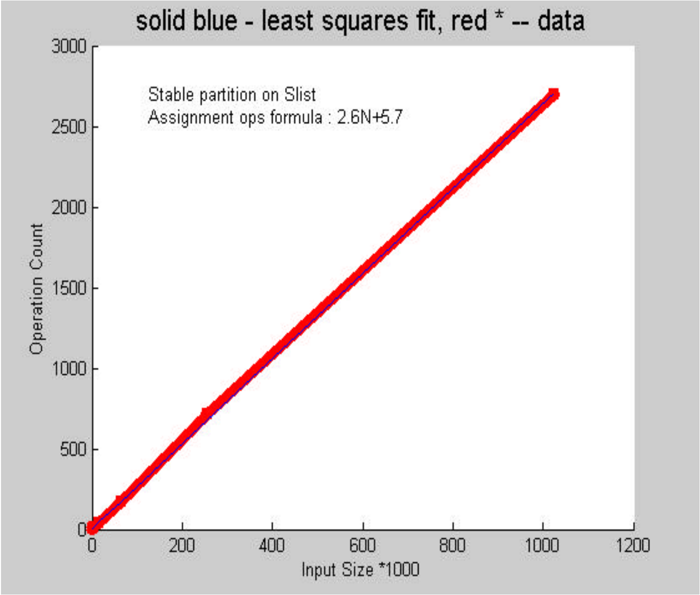


A.6.3 Total Operations

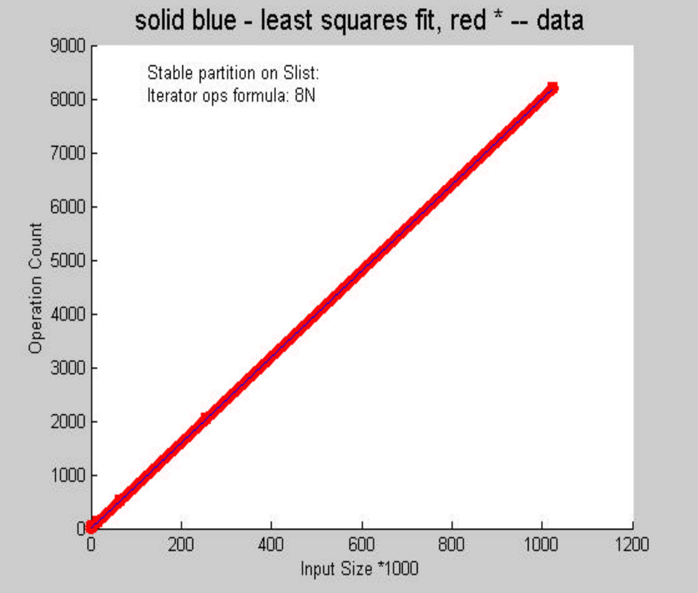


A.7 Slist - Stable Partition

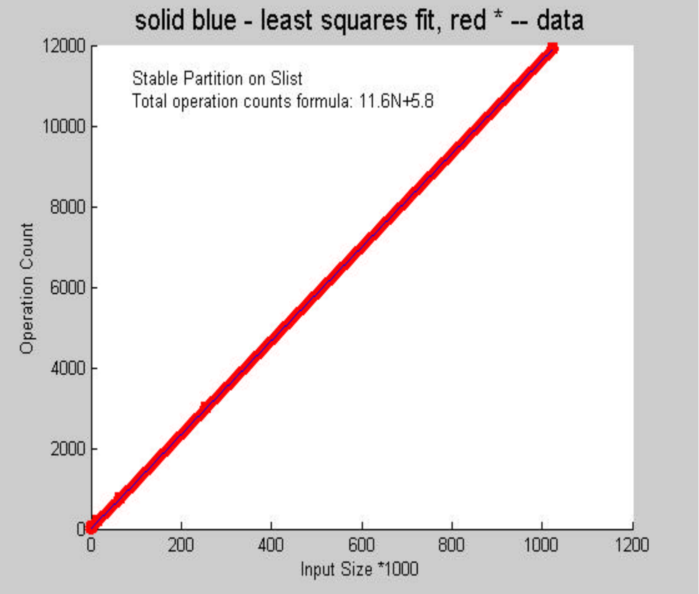
A.7.1 Value Assignments



A.7.2 Iterator Operations

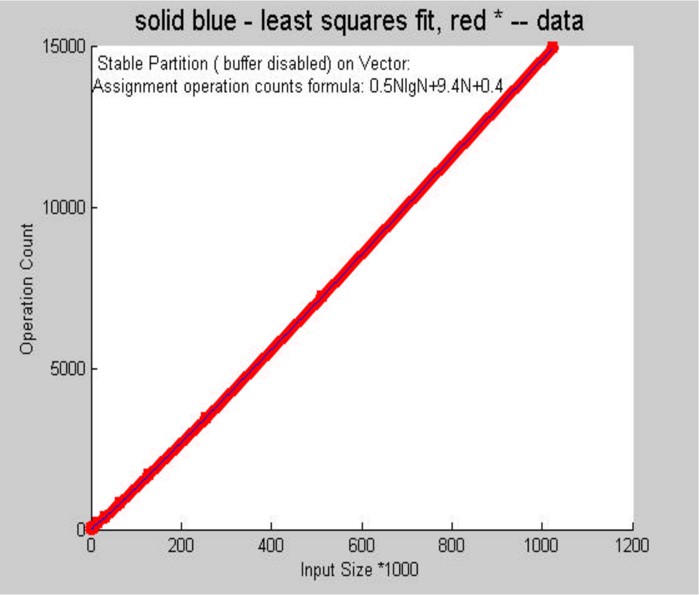


A.7.3 Total Operations

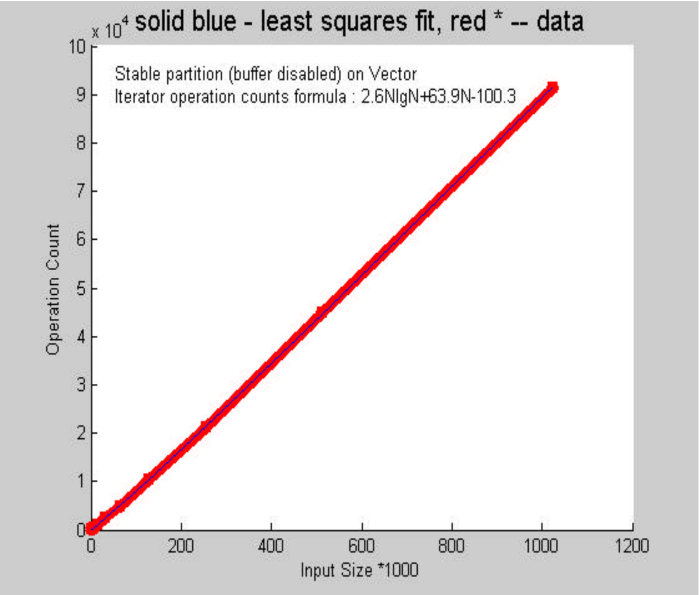


A.8 Worst Case

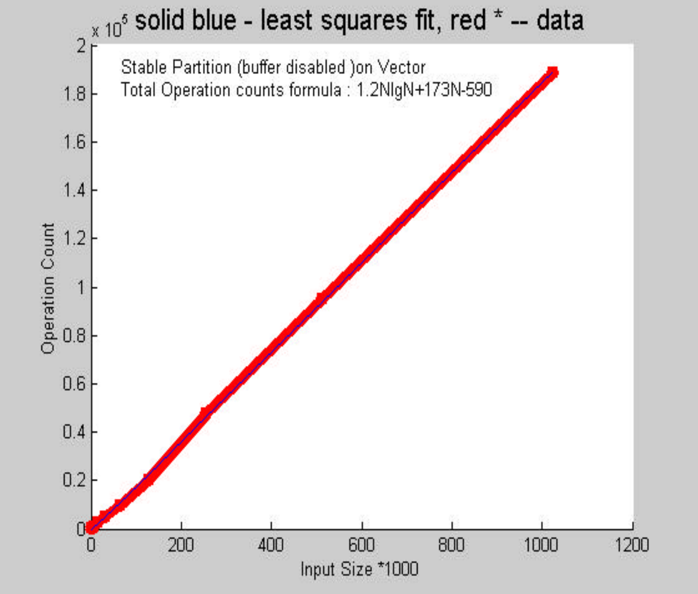
A.8.1 Value Assignments



A.8.2 Iterator Operations



A.8.3 Total Operations



References

[1] SGI Standard Template Library Reference
http://www.sgi.com/tech/stl

Table 3: Performance of Stable Partition on random sequences of vector type with and without buffer(sizes and operations counts in multiples of 1,000)

Size	With or without buffer	Comp- arisons	Assign- ments	Iterator Ops	Total Ops
1	With	1	2.434	6.024	9.463
	Without	1	9.554	54.447	119.278
2	With	2	5.304	12.024	19.333
	Without	2	21.223	125.27	288.717
4	With	4	10.553	24.024	38.582
	Without	4	43.253	281.011	584.511
8	With	8	21.524	48.024	77.553
	Without	8	86.063	508.634	1104.75
16	With	16	39.706	96.024	151.735
	Without	16	184.168	1091.24	2351.28
32	With	32	76.96	192.024	300.989
	Without	32	391.659	2339.66	5058.67
64	With	64	159.317	384.024	607.346
	Without	64	759.442	4727.14	9666.07
128	With	128	311.499	768.024	1207.53
	Without	128	1674.91	10291.6	20274.5
256	With	256	706.576	1536.02	2498.61
	Without	256	3451.6	21151.7	47407.7
512	With	512	1280.29	3072.02	4863.32
	Without	512	7214.78	450004.3	94830.7
1024	With	1024	2519.19	6144.02	9687.22
	Without	1024	14926.9	91489.8	188658