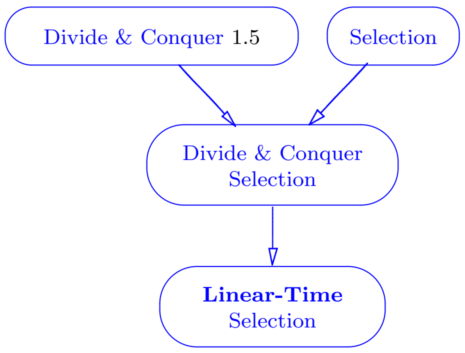
2.13 Worst-Case Linear Time Selection

Section authors: Geoff Greene, Seyit Camtepe, Armand Cistaro, and Sergej Roytman.



Refinement of: Divide & Conquer Selection, therefore of Selection and Divide & Conquer (§1.5)

Prototype:
```
int _selection(int *start,
               int *end,
               int *nth,
               unsigned int increment)
```

Input: An array, for which a sorted order exists, and an integer, which indicates a position in the array.

Output: The nth smallest element in the array, where n is the position indicated by `position`.

Effects:

- After execution, the elements in $[$`first`$,$ `last`$)$ are a permutation (§2.5) of the input.
- After execution the nth smallest element in the array is in position n. All elements smaller than the nth element are in positions lower than n, and all elements greater than the nth element are in positions higher than n.

Asymptotic complexity: Let $N = \texttt{last} - \texttt{first}$ (the number of elements in the array).

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

Complexity in terms of operation counts:

Average case operation counts for the linear time select on random data of size N

	Value operations		Iterator operations	
N	assignments	comparisons	movements	comparisons
1000	13136.40	7369.35	10741.30	13012.95
5000	70687.84	39796.18	57915.49	69544.89
10000	143604.19	80817.13	117797.32	141136.56
50000	726285.76	409010.72	595181.76	711749.08
100000	1469770.48	830991.70	1206920.62	1440846.72
500000	7371665.74	4164790.01	6048317.25	7217548.60
1000000	14789623.85	8358122.59	12138017.04	14475851.85
5000000	74223281.40	41986569.35	60935123.45	72627682.79

*N*th Element, Average-Case Results:

Operation counting adaptor library is used find iterator and value operation counters of STL *N*th Element.

	Value operations		Iterator operations	
N	assignments	comparisons	movements	comparisons
1000	1636	2558	1164	564
5000	7704	12740	5226	2592
10000	15497	24417	10428	5198
50000	75100	118830	50179	25061
100000	157213	260878	104904	52427
500000	818322	1413000	545686	272810
1000000	1765180	2889440	1176920	588433

STL *N*th Element, Worst-Case Results:

Operation counting adaptor library is modified to generate median-of-3 killer data sequence. Following table presents the iterator and value operation count values of STL *N*th Element.

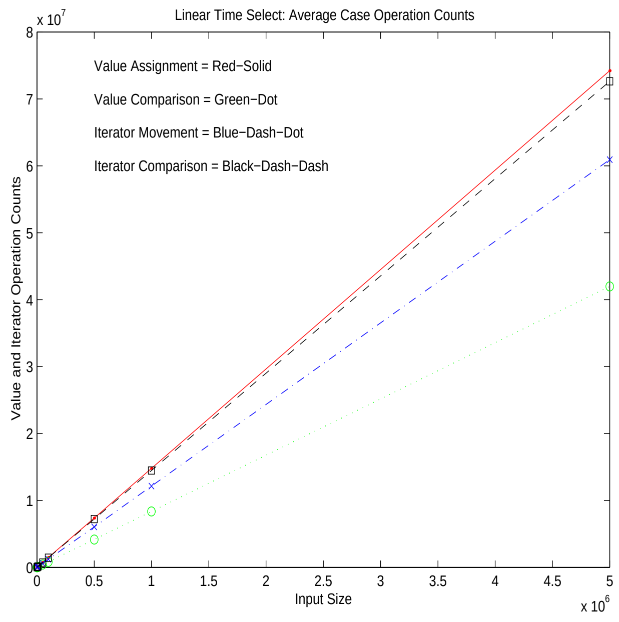
	Value operations		Iterator operations	
N	assignments	comparisons	movements	comparisons
1000	995	187738	1993	747
5000	4963	4668600	9929	3723
10000	9899	18631700	19801	7425
50000	49483	465545000	98969	37113

Formulas for Average Cases:

Following formulas are obtained by application of curve fitting function on operation count data.

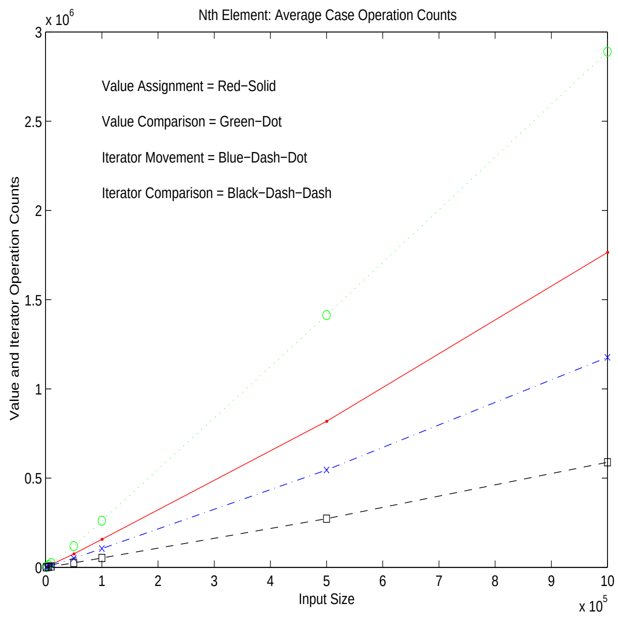
Linear Time Select:

Value assignments:	$14.7N - 3.257$
Value comparisons:	$8.3N - 2.246$
Iterator movements:	$12.1N - 2.867$
Iterator comparisons:	$14.4N - 2.987$



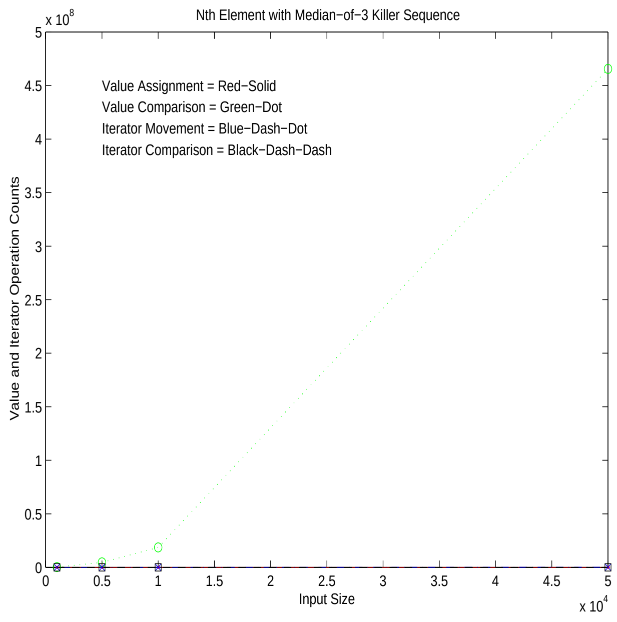
*N*th Element

Value assignments:	$1.51N + 527$
Value comparisons:	$2.8N - 7.31$
Iterator movements:	$N + 440$
Iterator comparisons:	$0.5N + 200$



Formulas for *N*th Element with median-of-3 Killer Sequence Input:

Value assignments:	$0.98N + 9.3$
Value comparisons:	$0.2N^2 + 5.8N - 3.776$
Iterator movements:	$1.9N + 2.4$
Iterator comparisons:	$0.7N + 0.5$



Block Size:

Select algorithm determines the *i*th smallest of an input array. It finds desired element by recursively partitioning the input array from a pivot element. Selection of pivot element is important to guarantee a good partitioning. Algorithm divides n elements of input array into $\lfloor n/5 \rfloor$ group of *block size* elements each and at most one group made up of the remaining *n mod 5* elements. It then sort these blocks with insertion sort algorithm and finds pivot which is medians of medians of these groups.

The effects of block size on algorithm is experimented. Results for three input sizes (1000, 100000, 1000000) are presented below. Block size of 7 appears to be best for minimizing total operation counts.

$n = 1,000$:

	Value operations		Iterator operations	
block size	assignments	comparisons	movements	comparisons
5	13395.9	7494.3	10963.0	13279.5
7	12705.6	7851.8	10431.2	12194.8
9	12786.7	8450.9	10627.1	12169.2
11	13117.2	9086.7	11013.4	12412.7
13	13798.2	9980.4	11760.1	13065.5
15	14772.1	11001.8	12698.5	14007.5
17	14396.3	10976.2	12486.7	13713.7
19	15399.7	11914.8	13411.9	14564.9
21	16211.2	12787.9	14232.3	15410.1
23	16879.3	13483.6	14902.0	15990.7

$n = 100,000$:

	Value operations		Iterator operations	
block size	assignments	comparisons	movements	comparisons
5	1460654.9	824043.5	1197572.9	1431357.3
7	1377539.3	860630.4	1139416.3	1319593.8
9	1374428.9	915387.9	1146809.4	1303529.2
11	1422009.8	997187.6	1200954.4	1345162.6
13	1487314.8	1080933.6	1269590.4	1405561.0
15	1570683.9	1181099.1	1356403.8	1486629.9
17	1649342.7	1270335.7	1436750.9	1563510.5
19	1690214.6	1319828.6	1479128.4	1602166.0
21	1790746.0	1427606.6	1581448.6	1701977.0
23	1878436.4	1520407.0	1669554.2	1788296.8

$n = 1,000,000$:

	Value operations		Iterator operations	
block size	assignments	comparisons	movements	comparisons
5	14770328.2	8348604.8	12121340.5	14459619.5
7	13724134.2	8566152.0	11333470.7	13134565.3
9	13907988.5	9305382.2	11623458.0	13194816.1
11	14294619.4	10029237.8	12073641.2	13517942.2
13	14765857.2	10711782.1	12585125.2	13947539.1
15	15702338.6	11787946.3	13548896.2	14854898.0
17	16392011.7	12588612.5	14259845.5	15525419.1
19	17072394.0	13356668.6	14955133.1	16189517.2
21	17983731.4	14340509.8	15879893.6	17088418.4
23	18717939.6	15130228.4	16623638.3	17812369.5

Linear Time Select Animation