

Searching Algorithms:

Sequential Search: Examine every item in the list until you find the value you're looking for.

Complexity Class: $O(N)$

The example to the right shows the steps to finding 3 in a list of integers.

22	1	6	3	-15	17
----	---	---	---	-----	----

22	1	6	3	-15	17
----	---	---	---	-----	----

22	1	6	3	-15	17
----	---	---	---	-----	----

22	1	6	3	-15	17
----	---	---	---	-----	----

Binary Search—Complexity Class: $O(\log N)$

* Only works if the list is sorted

1. Compare the element at the middle position in the list to the target value.
2. If the target value is equal to the element at the middle position, then you are done.
3. If the target value is less than the element at the middle position, then repeat the procedure starting from step 1 for the left half of the list.
4. If the target value is greater than the element at the middle position, then repeat the procedure starting from step 1 for the right half of the list.

Note: If either the left or right sides of the list are empty for steps 3 or 4, then the target value is not contained in the list.

Target Value: 9

Index	0	1	2	3	4	5	6	7	8
	-3	6	9	12	15	18	21	24	27

↑
LOW

↑
MID

↑
HIGH

Since 9 is greater than -3 and less than 15 and the list is sorted, we know 9 can't possibly be in the second half of the list. So we only continue searching in the first half.

Index	0	1	2	3	4	5	6	7	8
	-3	6	9	12	15	18	21	24	27

↑
LOW

↑
MID

↑
HIGH

9 is greater than 6 but less than 12, so we continue searching in the second half of the list.

Index	0	1	2	3	4	5	6	7	8
	-3	6	9	12	15	18	21	24	27

↑
LOW

↑
MID

↑
HIGH

We found 9!

Sorting Algorithms:

	Complexity	The Steps	Visual Representation																																																																																																				
Selection	$O(N^2)$	<ol style="list-style-type: none"> Look through the entire list for the smallest value. Swap the smallest value with the value at the current index (Unless current index contains the smallest value). Increase current index. Look through the rest of the list for the smallest value. Swap this value with the value at current index. Repeat for the rest of the list. 	<p>(Shaded boxes indicate swapped values)</p> <table border="1" data-bbox="1010 345 1961 477"> <tr><th>Index</th><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td></tr> <tr><td></td><td>7</td><td>4</td><td>2</td><td>16</td><td>22</td><td>13</td><td>15</td><td>31</td><td>0</td></tr> </table> <p style="text-align: center;"> ↑ Current Index ↑ Smallest Value </p> <table border="1" data-bbox="1010 591 1961 722"> <tr><th>Index</th><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td></tr> <tr><td></td><td>0</td><td>4</td><td>2</td><td>16</td><td>22</td><td>13</td><td>15</td><td>31</td><td>7</td></tr> </table> <p style="text-align: center;"> ↑ Current Index ↙ Smallest Value </p> <table border="1" data-bbox="1010 836 1961 967"> <tr><th>Index</th><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td></tr> <tr><td></td><td>0</td><td>2</td><td>4</td><td>16</td><td>22</td><td>13</td><td>15</td><td>31</td><td>7</td></tr> </table> <p style="text-align: center;"> ↑ Current Index In this case, 4 is the smallest value so we don't need to swap anything. </p> <table border="1" data-bbox="1010 1081 1961 1213"> <tr><th>Index</th><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td></tr> <tr><td></td><td>0</td><td>2</td><td>4</td><td>16</td><td>22</td><td>13</td><td>15</td><td>31</td><td>7</td></tr> </table> <p style="text-align: center;"> ↑ Current Index ↑ Smallest Value </p> <table border="1" data-bbox="1010 1326 1961 1458"> <tr><th>Index</th><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td></tr> <tr><td></td><td>0</td><td>2</td><td>4</td><td>7</td><td>22</td><td>13</td><td>15</td><td>31</td><td>16</td></tr> </table> <p>This process continues until you've reached the end of the list.</p>	Index	0	1	2	3	4	5	6	7	8		7	4	2	16	22	13	15	31	0	Index	0	1	2	3	4	5	6	7	8		0	4	2	16	22	13	15	31	7	Index	0	1	2	3	4	5	6	7	8		0	2	4	16	22	13	15	31	7	Index	0	1	2	3	4	5	6	7	8		0	2	4	16	22	13	15	31	7	Index	0	1	2	3	4	5	6	7	8		0	2	4	7	22	13	15	31	16
Index	0	1	2	3	4	5	6	7	8																																																																																														
	7	4	2	16	22	13	15	31	0																																																																																														
Index	0	1	2	3	4	5	6	7	8																																																																																														
	0	4	2	16	22	13	15	31	7																																																																																														
Index	0	1	2	3	4	5	6	7	8																																																																																														
	0	2	4	16	22	13	15	31	7																																																																																														
Index	0	1	2	3	4	5	6	7	8																																																																																														
	0	2	4	16	22	13	15	31	7																																																																																														
Index	0	1	2	3	4	5	6	7	8																																																																																														
	0	2	4	7	22	13	15	31	16																																																																																														

AP Computer Science A Searching and Sorting Algorithms Cheat Sheet

Mergesort	$O(N \log N)$	<ol style="list-style-type: none">1. Repeatedly divide the list into two equal parts until each part is a single element of the list2. Combine the parts in sorted order, until the list is completely reconstructed.	<p>The diagram illustrates the Merge Sort algorithm. It starts with an array of four elements: 8, 0, 7, and 4. The process is shown in two columns, representing the recursive division and the subsequent merging and sorting.</p> <p>Division Phase:</p> <ul style="list-style-type: none">Initial array: [8, 0, 7, 4]Step 1: Divide into two halves: [8, 0] and [7, 4]Step 2: Divide each half into single elements: [8], [0], [7], and [4] <p>Combine and Sort Phase:</p> <ul style="list-style-type: none">Step 3: Merge and sort the first half: [0, 8]Step 4: Merge and sort the second half: [4, 7]Step 5: Final merge and sort: [0, 4, 7, 8]
-----------	---------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

AP Computer Science A Searching and Sorting Algorithms Cheat Sheet

<p>Insertion</p>	<p>$O(N^2)$</p>	<ol style="list-style-type: none"> 1. Divide list into two imaginary lists: sorted (initially empty) and unsorted (the rest of the elements). 2. Take the first element from the unsorted list and place into sorted. 3. Take the next element from the unsorted list and insert the value into the correct location. 4. Repeat until the unsorted part is empty. 	<div style="display: flex; justify-content: space-around;"> <div style="text-align: center;"> <p>Unsorted List</p> <table border="1" style="margin: 0 auto;"> <tr><td>4</td><td>-1</td><td>0</td><td>13</td><td>8</td><td>5</td></tr> </table> </div> <div style="text-align: center;"> <p>Sorted List</p> <table border="1" style="margin: 0 auto;"> <tr><td> </td><td> </td><td> </td><td> </td><td> </td><td> </td></tr> </table> </div> </div> <p>Move the first value into the sorted list</p> <div style="display: flex; justify-content: space-around;"> <table border="1" style="margin: 0 auto;"> <tr><td>-1</td><td>0</td><td>13</td><td>8</td><td>5</td></tr> </table> → <table border="1" style="margin: 0 auto;"> <tr><td>4</td><td> </td><td> </td><td> </td><td> </td><td> </td></tr> </table> </div> <p>-1 is less than 4, so we insert it in front of 4</p> <div style="display: flex; justify-content: space-around;"> <table border="1" style="margin: 0 auto;"> <tr><td>0</td><td>13</td><td>8</td><td>5</td></tr> </table> → <table border="1" style="margin: 0 auto;"> <tr><td>-1</td><td>4</td><td> </td><td> </td><td> </td><td> </td></tr> </table> </div> <p>0 should be inserted between -1 and 4</p> <div style="display: flex; justify-content: space-around;"> <table border="1" style="margin: 0 auto;"> <tr><td>13</td><td>8</td><td>5</td></tr> </table> → <table border="1" style="margin: 0 auto;"> <tr><td>-1</td><td>0</td><td>4</td><td> </td><td> </td><td> </td></tr> </table> </div> <div style="display: flex; justify-content: space-around;"> <table border="1" style="margin: 0 auto;"> <tr><td>8</td><td>5</td></tr> </table> → <table border="1" style="margin: 0 auto;"> <tr><td>-1</td><td>0</td><td>4</td><td>13</td><td> </td><td> </td></tr> </table> </div> <div style="display: flex; justify-content: space-around;"> <table border="1" style="margin: 0 auto;"> <tr><td>5</td></tr> </table> → <table border="1" style="margin: 0 auto;"> <tr><td>-1</td><td>0</td><td>4</td><td>8</td><td>13</td><td> </td></tr> </table> </div> <div style="text-align: center; margin-top: 10px;"> <p>Final, Sorted List</p> <table border="1" style="margin: 0 auto;"> <tr><td>-1</td><td>0</td><td>4</td><td>5</td><td>8</td><td>13</td></tr> </table> </div>	4	-1	0	13	8	5							-1	0	13	8	5	4						0	13	8	5	-1	4					13	8	5	-1	0	4				8	5	-1	0	4	13			5	-1	0	4	8	13		-1	0	4	5	8	13
4	-1	0	13	8	5																																																													
-1	0	13	8	5																																																														
4																																																																		
0	13	8	5																																																															
-1	4																																																																	
13	8	5																																																																
-1	0	4																																																																
8	5																																																																	
-1	0	4	13																																																															
5																																																																		
-1	0	4	8	13																																																														
-1	0	4	5	8	13																																																													